ENGENHARIA DE COMPUTAÇÃO

Cassiano de Mello Padilha

AVALIAÇÃO EXPERIMENTAL DE TECNOLOGIAS SDN PARA IMPLANTAÇÃO EM REDES DE PRODUÇÃO

Santa Cruz do Sul
2017

ENGENHARIA DE COMPUTAÇÃO

Cassiano de Mello Padilha

AVALIAÇÃO EXPERIMENTAL DE TECNOLOGIAS SDN PARA IMPLANTAÇÃO EM REDES DE PRODUÇÃO

Prof. Me. Lucas Fernando Muller
Orientador

Santa Cruz do Sul
2017

ENGENHARIA DE COMPUTAÇÃO

Cassiano de Mello Padilha

AVALIAÇÃO EXPERIMENTAL DE TECNOLOGIAS SDN PARA IMPLANTAÇÃO EM REDES DE PRODUÇÃO

Prof. Me. Charles Varlei Neu

Avaliador

Santa Cruz do Sul

2017

ENGENHARIA DE COMPUTAÇÃO

Cassiano de Mello Padilha

AVALIAÇÃO EXPERIMENTAL DE TECNOLOGIAS SDN PARA IMPLANTAÇÃO EM REDES DE PRODUÇÃO

Prof. Me. Ricardo Mello Czekster

Avaliador

Santa Cruz do Sul

2017

Cassiano de Mello Padilha

AVALIAÇÃO EXPERIMENTAL DE TECNOLOGIAS SDN PARA IMPLANTAÇÃO EM REDES DE PRODUÇÃO

Trabalho de Conclusão II apresentado ao Curso de Engenharia de Computação da Universidade de Santa Cruz do Sul, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Lucas Fernando Muller

Santa Cruz do Sul

2017

"Consulte não a seus medos, mas a suas esperanças e sonhos. Pense não sobre suas frustrações, mas sobre seu potencial não usado. Preocupe-se não com o que você tentou e falhou, mas com aquilo que ainda é possível a você fazer. ”

— PAPA JOÃO XXIII

RESUMO

Redes Definidas por Software é um paradigma emergente que propõe a separação do plano de controle e do plano de dados, buscando tornar os elementos de encaminhamento mais simples, enquanto o processamento da lógica de encaminhamento será realizado por um elemento externo denominado controlador. Devido a essa separação, diversos modelos de controladores SDN (*Software Defined Networking*) foram desenvolvidos, assim, há questões sobre desempenho, escalabilidade, segurança e confiabilidade que carecem de respostas para auxiliar na escolha da correta plataforma de controle, tornando-se assim uma etapa complexa. Dessa forma, realizar avaliações através de ferramentas de *benchmarking* é algo que deve preceder a implantação de uma SDN. Entretanto, atualmente as ferramentas que têm esse propósito possuem diversas limitações, principalmente por não terem padrões definidos para experimentos e assim deixarem lacunas que inviabilizam algumas análises importantes. Com base nesse contexto, esse trabalho de conclusão propõe uma avaliação sistemática de um conjunto de duas ferramentas que possuem o propósito de avaliar controladores SDN, e então verificar suas vantagens e desvantagens e, posteriormente, tomando como base metodologias recentes definidas para avaliação de controladores, elaborar um protótipo de ferramenta que contribua para a análise das plataformas de controle, como também subsidie a escolha do modelo correto.

Palavras-chave: Benchmarking, Controlador, Ferramenta, Implantação, SDN.

ABSTRACT

Software Defined Networking is an emerging paradigm that proposes the separation of the control plan and the data plan, seeking to make the routing elements simpler, while the processing of the routing logic will be executed by an external element called controller. Due to this separation, several models of SDN controllers have been developed, so there are issues about performance, scalability, security and reliability issues that require answers to assist in choosing the right control platform, making it a difficult step. Therefore, evaluating benchmarking tools is something that must precede the deployment of an SDN Network. However, currently, the tools that have this purpose have several limitations, mainly because they do not have defined standards for experiments and thus leave gaps that make some important analyzes unfeasible. Based on this context, this work proposes a systematic evaluation of a set of two tools that have the purpose of evaluating SDN controllers. The evaluation will check their advantages and disadvantages and, later, based on recent methodologies defined for the evaluation of controllers, devolop a tool that contribute to the analysis of the control platforms, as well as support the choice of the correct model.

Keywords: Benchmarking, Controllers, Tool, Deployment, SDN.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface
APP	Application
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
FIB	Forwarding Information Base
Gb	Gigabit
GHz	Gigahertz
IP	Internet Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
LTS	Long Term Support
MAC	Media Access Control
OC	OFCBenchmark Client
OCC	OFCBenchmark Control Center
ONF	Open Networking Foundation
ONOS	Open Network Operating System
OVS	Open Virtual Switch
RAM	Random Access Memory
RFC	Request for Comments
RIB	Routing Information Base
RTT	Round-Trip Time
SDN	Software-Defined Networking
SSL	Secure Socket Layer
SYNC	Synchronization
TCP	Transmission Control Protocol
UNISC	Universidade de Santa Cruz do Sul
VLAN	Virtual Lan
VM	Virtual Machine
VS	Virtual Switch

Sumário

1 INTRODUÇÃO

Redes Definidas por Software (*Software Defined Networking* - SDN) é um paradigma emergente de redes de computadores que tem como objetivo principal facilitar a gerência de operações na rede, habilitando a inovação e simplificando as atividades de controle [KREUTZ *et al.,* 2014]. Propondo assim, a separação do plano de controle da rede do plano de dados. Dessa forma, um dos objetivos deste novo paradigma é tornar os elementos de encaminhamento (*switches* e roteadores) mais simples, enquanto o processamento da lógica de encaminhamento será realizado por um novo elemento introduzido na rede, o controlador ou sistema operacional de rede [KREUTZ *et al.,* 2014].

Devido ao desacoplamento que ocorre entre o plano de controle e o plano de dados, um protocolo faz-se necessário para que ocorra a comunicação entre eles. Nesta direção, SDN adota o protocolo OpenFlow [ONF, 2016], o qual é definido pela *Open Networking Foundation*. O protocolo OpenFlow fornece as abstrações necessárias para programação dos dispositivos de encaminhamento da rede, sendo atualmente a implementação mais significativa deste novo modelo de rede [MCKEOWN *et al.,* 2008]. O controlador da rede, nesse caso, atua como uma entidade logicamente centralizada, que possui uma visão global da infraestrutura e gerencia um conjunto distribuído e programável de dispositivos de encaminhamento de pacotes [NUNES *et al.,* 2014].

A ideia de redes programáveis foi proposta como uma forma de garantir e facilitar a evolução da rede. Garantindo maior independência aos desenvolvedores de software e permitindo, em particular, uma gestão mais simplificada da rede. O campo de Redes Definidas por Software é bastante recente tendendo a crescer em um ritmo muito rápido [NUNES *et al.,* 2014]. Conforme evidenciado no estado-da-arte, há um crescente interesse em usufruir dos potenciais benefícios de SDN [LEVIN *et al.,* 2014; KANDULA *et al.,* 2013; JAIN *et al.,* 2013]. Para isso temos de enfrentar um desafio fundamental que ocorre no momento da implantação[VISSICCHIO *et al.,* 2014]. SDN apresenta seu próprio conjunto de desafios e limitações, passando por dificuldades na implantação até a busca por garantias relacionadas a resiliência e escalabilidade, relacionadas à lógica de controle centralizada. Tal situação tem levado a comunidade acadêmica e a indústria a estudar métodos de migração de forma que os desafios sejam minimizados [LEVIN *et al.,* 2014; VISSICCHIO *et al.,* 2014].

Desse modo, uma etapa importante a fim de guiar as ações que devem ser tomadas sobre a infraestrutura é a análise das diferentes tecnologias relacionadas ao projeto de migração. Enquanto o controlador certamente preenche o papel de um sistema operacional tradicional,

fazendo a ponte entre o hardware e as aplicações, muitos destes modelos desenvolvidos não fornecem uma estabilidade e desempenho que seria de se esperar de um sistema operacional [JARSCHEL *et al.*, 2012], em particular, no que se refere ao momento de implantação, há questões sobre o desempenho, robustez, confiabilidade e conformidade com especificações, que carecem de respostas para subsidiar a tomada de decisão [SEZER *et al.*, 2013]. Uma vez que o padrão OpenFlow não determina como cada controlador deve ser implementado, os diferentes modelos de controladores podem ser específicos para determinados cenários de rede. Desta forma, tais respostas só podem ser obtidas através de uma adequada avaliação, uma vez que não são encontradas todas as respostas no estado-da-arte.

Atualmente, a literatura sobre SDN apresenta poucas soluções em torno de avaliação experimental de tecnologias SDN para implantação em redes de produção. Ferramentas que possuem o objetivo de avaliar controladores SDN foram desenvolvidas, entretanto muitas delas possuem limitações ou resultados pouco expressivos para auxiliar nessa etapa. Fato este que acaba por gerar dificuldades no processo de implantação quando lembramos que a escolha de uma plataforma de controle é uma das primeiras escolhas (além da escolha dos dispositivos de encaminhamento) a ser realizada que irá influenciar no projeto final da rede SDN.

Neste contexto, para este trabalho de conclusão de curso, buscou-se efetuar uma avaliação sistemática em torno das plataformas de controle SDN, propondo assim, especificamente, realizar avaliações através de um conjunto de duas ferramentas de *benchmark*. Desse modo, teve-se como objetivo principal obter resultados mais detalhados sobre os possíveis gargalos de desempenho de cada modelo de controlador, bem como seu comportamento e desempenho em uma rede OpenFlow. E ao final, após verificar as vantagens e desvantagens dessas ferramentas, foi desenvolvido um protótipo com base em metodologias recentes definidas pela entidade IETF (*Internet Engineering Task Force)* para avaliação de controladores, buscando assim contribuir para a análise das plataformas de controle.

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2, são apresentados os fundamentos para a realização deste trabalho, abordando os conceitos referentes à SDN. No Capítulo 3, são abordados conceitos sobre controladores OpenFlow. No Capítulo 4, apresenta-se o estado da arte, abordando os trabalhos relacionados e implementações. No Capítulo 5 são apresentadas análises de duas ferramentas de *benchmark* de controladores OpenFlow. Já no Capítulo 6, é apresentado o protótipo da ferramenta de *benchmark* desenvolvida. No Capítulo 7 são definidas as metodologias de avaliação do protótipo desenvolvido, abordando o que será avaliado, ambiente de avaliação e cenário de

avaliações. No Capítulo 8 são descritos os resultados obtidos através dos experimentos realizados. Por fim, no Capítulo 9, as considerações finais e trabalhos futuros.

2 REDES DEFINIDAS POR SOFTWARE

Neste capítulo serão apresentados os principais conceitos para o desenvolvimento deste trabalho de conclusão. Inicialmente, na Seção 2.1, são abordados os conceitos básicos relativos à visão geral do paradigma SDN, mostrando principalmente como ocorreu a exploração desse conceito, como também aspectos relativos à sua operação. Em seguida, na Seção 2.2, é apresentada uma visão geral sobre o protocolo OpenFlow, como ocorreu seu surgimento e como atua no gerenciamento de uma Rede Definida por Software. Por fim, na Seção 2.3, são apresentadas algumas considerações finais referente ao paradigma SDN.

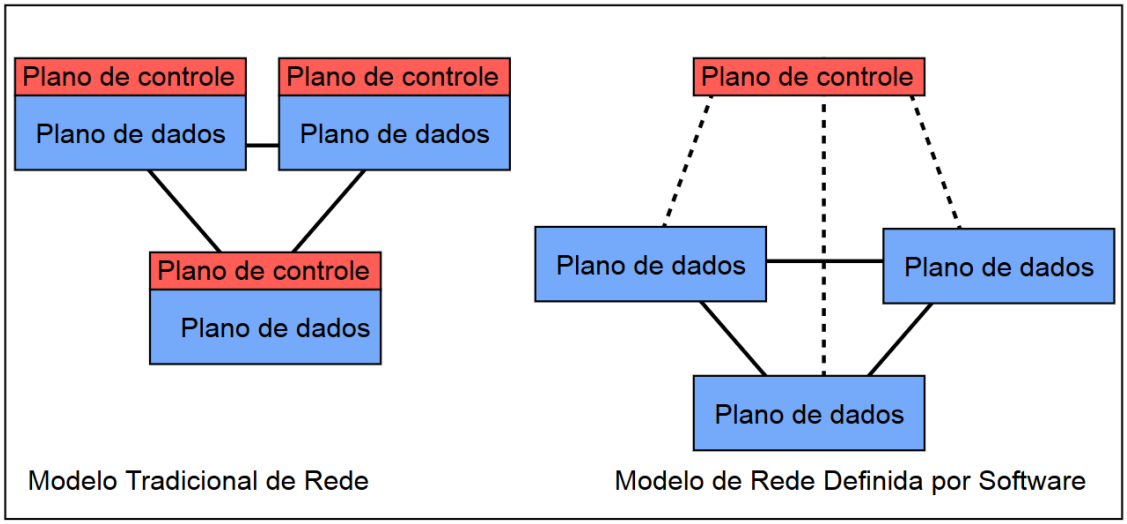
2.1 Visão Geral

Conforme apresentado por Nunes *et al.,* [2014] o modelo de Redes Definidas por Software representa um conceito novo na área de redes de computadores e, devido a todo sucesso, em pesquisas acadêmicas, vem sendo adotado no atual mercado de redes de computadores. Hoje em dia, grandes empresas produtoras de equipamentos de rede já incorporam em seus produtos soluções que abordam o uso de SDN, com a ideia de redes programáveis [KREUTZ *et al.,*2014]. Esta abordagem de redes programáveis tem sido proposta como uma forma de facilitar drasticamente a gestão da rede, permitindo assim maiores inovações, de modo que através do seu uso a rede não fique dependente e presa aos diversos protocolos TCP na camada de transporte e ao IP na camada de rede, limitando dessa forma a expansão da Internet [KREUTZ *et al.,*2014]. Em particular, SDN é um novo paradigma de rede em que o hardware de encaminhamento de dados é dissociado das decisões de controle, permitindo, através desse novo modelo, que desenvolvedores de softwares possam obter maior controle sobre os recursos da rede [Nunes *et al.,* 2014].

Em SDN, a inteligência da rede está logicamente centralizada em controladores baseados em software (plano de controle) e, através dos dispositivos de encaminhamento da rede (como *switches* e roteadores) o encaminhamento de pacotes torna-se mais simples. Ao contrário do que ocorre atualmente nos modelos de rede IP tradicionais, em que os planos de dados e de controles estão acoplados em um mesmo equipamento, impossibilitando assim abordagens mais complexas, como a criação de redes com propósitos mais específicos. Esse novo modelo de rede foi desenvolvido principalmente com a ideia de permitir um simples e programático controle de encaminhamento de dados na rede. Como visualizado na Figura 1, a separação do hardware de encaminhamento da lógica de controle facilita a implantação de novos protocolos e aplicações, permitindo assim uma visualização direta da rede e um gerenciamento consolidado. Assim, como evidenciado por Nunes *et al.,* [2014], ao invés de impor políticas de

funcionamento sobre os protocolos, a rede é reduzida em: encaminhamento de hardware "simples" e controlador de rede responsável pela tomada de decisões.

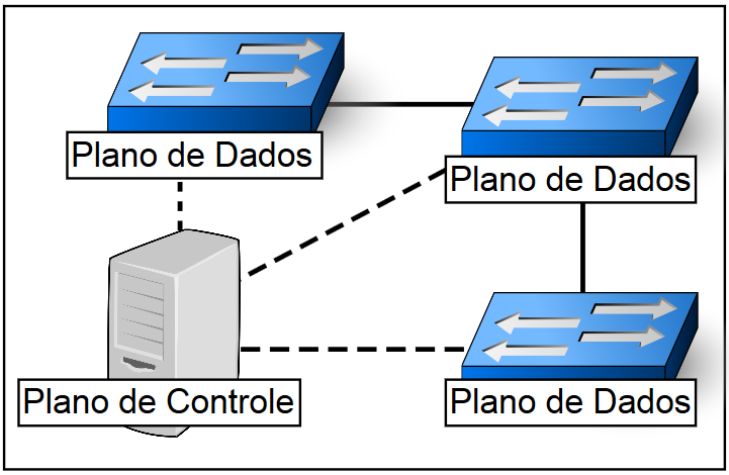
Figura 1 - Comparação entre o modelo tradicional de rede e o modelo SDN.
Fonte: Adaptado de Nunes *et al.*, [2014].



Baseado neste contexto, a arquitetura SDN separa os dispositivos de rede em duas partes: plano de dados e plano de controle, onde essa separação é a chave principal para a flexibilidade almejada na rede. Sendo assim, é possível representar o plano de dados como os equipamentos de encaminhamento de dados na rede, como *switches* e roteadores. Entretanto, estes equipamentos são responsáveis apenas pelo encaminhamento de dados dos usuários e por garantir suporte necessário para conectividade e segurança [ONF, 2016].

Em conjunto com o plano de dados, o plano de controle representa os protocolos ou serviços de rede utilizados para preencher as tabelas de encaminhamento dos elementos do plano de dados. Onde através dessa separação, o controle da rede fica em torno de um elemento externo, logicamente centralizado, denominado de controlador. O controlador possui uma visão global de toda a rede, ficando assim responsável por gerenciar as rotas a serem incorporadas nos equipamentos da rede. Dessa maneira, com esta entidade centralizada, o desenvolvimento de novos serviços de rede torna-se mais simples, e qualquer modificação ou adição de novas rotas só precisam ser configuradas neste único ponto. A relação entre o plano de dados e o plano de controle é ilustrada na Figura 2.

Figura 2 - Arquitetura SDN.
Fonte: do autor.



De acordo com McKeown *et al.*, [2008], o padrão SDN foi realmente definido como um modelo de rede após a padronização do protocolo OpenFlow, que surgiu para criar uma interface entre o plano de controle e o plano de dados. Possibilitando, através dessa interface, uma comunicação segura entre os diversos equipamentos da rede e o controlador. O protocolo OpenFlow não é o único protocolo desenvolvido até hoje, mas sim aquele que possui maior estudo e investimento, além de estar em um estado mais avançado, proporcionando assim maior confiança na comunicação através de uma interface aberta e padronizada [ONF, 2016].

Nesta direção, o conceito por trás do paradigma SDN vem evoluindo cada vez mais, impulsionado principalmente pelo desejo de proporcionar uma gestão controlada da rede. Assim, com a consolidação do padrão de protocolo OpenFlow, pode-se trazer a implementação de modelos SDN para próximo da realidade, deixando de ser assuntos tratados apenas na academia, para se tornar uma realidade em torno dos novos e modernos dispositivos de rede.

2.2 Protocolo OpenFlow

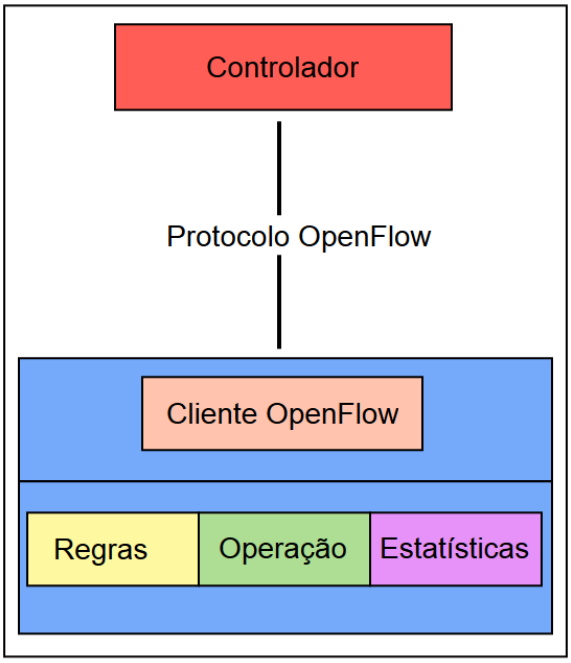
O protocolo OpenFlow surgiu com a necessidade de se definir uma interface entre os dispositivos da rede (*switches* e roteadores) e o controlador, tornando-se assim o marco da difusão do paradigma SDN. Dessa forma, o OpenFlow constitui-se de uma arquitetura aberta e programável, sendo implementado em ambos os planos da rede, garantindo assim a interoperabilidade com os diversos modelos de dispositivos de rede, bem como prevendo um canal seguro para comunicação entre os dispositivos de encaminhamento e o controlador [MCKEOWN *et al.*, 2008]. O OpenFlow, atualmente, conta com atualizações constantes, sendo mantido e atualizado até a versão atual (1.5.1) pela *Open Network Foundation* [ONF, 2016].

Através da arquitetura OpenFlow, tem-se uma grande vantagem que é a possibilidade oferecida por ela de se desenvolver de maneira independente o tratamento dos fluxos de dados, definindo assim como estes dados devem ser distribuídos pela rede. À vista disso, o OpenFlow

fornece um protocolo aberto para programar a tabela de fluxo em diferentes *switches* e roteadores. Desse modo, possibilitando para os administradores de rede dividir o tráfego em fluxos de produção e de pesquisa, isto é, pesquisadores tem o controle de seus próprios fluxos na rede, escolhendo as rotas que seus pacotes irão seguir, bem como o tratamento que irão receber [MCKEOWN *et al.*, 2008].

Baseado neste contexto, na arquitetura do protocolo OpenFlow, ilustrada na Figura 3, os dispositivos da rede, no caso um *switch* OpenFlow, contém uma ou mais tabelas de fluxo e uma camada de abstração responsável por efetuar a comunicação de forma segura com um controlador. Desse modo, a comunicação entre o controlador e o dispositivo de encaminhamento acontece através do protocolo OpenFlow, que define um conjunto de mensagens que podem ser trocadas entre essas entidades. Dessa maneira, permitindo que um controlador remoto possa, por exemplo, adicionar ou apagar fluxos de entrada nas tabelas de fluxo dos *switches*.

Figura 3 - Arquitetura do Prototolo OpenFlow.
Fonte: Adaptado de Nunes *et al.*, [2014].



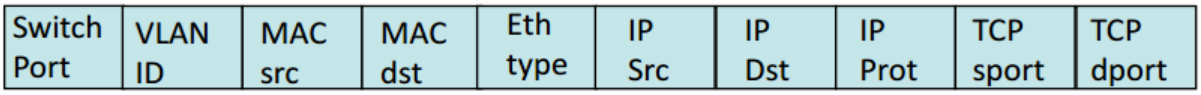
Com base no modelo OpenFlow, tem-se como uma principal característica a separação do plano de dados e o plano de controle. O plano de dados de um *switch* OpenFlow consiste, basicamente, de uma tabela de fluxo e uma ação associada a cada entrada na tabela. Desse modo, um *switch* OpenFlow é dividido basicamente em três partes, conforme ilustrado na Figura 3 [MCKEOWN *et al.*, 2008]:

I. **Tabela de fluxos:** Possui uma ação associada a cada entrada de fluxo, respondendo ao *switch* como processar cada fluxo. Essa tabela de fluxo é composta por uma sequência de campos de cabeçalhos, como ilustra a figura 4. Desse modo, quando um pacote chega no *switch*, este verifica primeiramente seu cabeçalho, identificando assim se já existe na tabela de fluxo, caso contrário o encapsula e envia para o controlador, para que este defina as regras e o adicione na tabela.

II. **Canal seguro:** Responsável por intermediar a comunicação entre o *switch* e o controlador, permitindo que comandos e pacotes possam ser transmitidos entre eles de maneira segura, geralmente através de um padrão de segurança, como por exemplo o *Secure Socket Layer* (SSL).

III. **Protocolo OpenFlow:** Proporciona uma interface padrão, que permite a comunicação entre os *switches* e o controlador.

Figura 4 - Cabeçalho de uma mensagem do protocolo OpenFlow.
Fonte: ONF, [2016].



Neste mesmo contexto, com base nos *switches* OpenFlow, com relação aos fluxos de entradas, estes dispositivos de rede devem executar três ações básicas [MCKEOWN *et al.*, 2008]:

I. Encaminhar pacotes do fluxo de entrada para uma determinada porta, de acordo com uma regra pré-estabelecida na tabela de fluxo definida pelo controlador.

II. Encapsular e encaminhar através de um canal seguro pacotes do fluxo de entrada para um controlador, de modo que o controlador decida se o fluxo deve ou não ser adicionado na tabela de fluxo.

III. Descartar pacotes do fluxo de entrada, servindo em alguns casos como uma solução para segurança, reduzindo assim, por exemplo, ataques de negação de serviço (DoS – *Denial of Service*).

Para facilitar a implementação da comunicação entre os dispositivos de encaminhamento e o controlador, o protocolo OpenFlow fornece três tipos distintos de troca de mensagens: (i) mensagens do controlador para o *switch*. (ii) mensagens assíncronas. (iii) mensagens simétricas [ONF, 2016]. Cada uma específica para determinada situação. Dessa forma, mensagens do tipo controlador para o *switch* são iniciadas pelo controlador e utilizadas para gerenciar ou inspecionar o estado do *switch* diretamente, e são encontradas sete mensagens distintas:

I. **Features:** o controlador solicita que o *switch* informe suas características físicas.

II. **Configuration:** o controlador define parâmetros de configuração em um *switch*.
III. **Modify-state:** esse tipo de mensagem é enviado para gerenciar o estado nos *switches*, onde sua principal finalidade é adicionar, excluir ou modificar entradas nas tabelas de fluxo.
IV. **Read-state:** usadas para coletar informações do *switch*, como configurações, estatística e capacidades.
V. **Packet-out:** utilizadas para enviar pacotes para uma porta específica do *switch.*
VI. **Barrier:** utilizadas pelo controlador para garantir se as dependências de mensagens foram cumpridas ou não.
VII. **Role-Request:** utilizadas pelo controlador para verificar seu papel no canal OpenFlow, ou consultar este papel.

Mensagens assíncronas são enviadas pelos *switches* sem a solicitação do controlador. Assim, *switches* enviam mensagens assíncronas ao controlador para alertar sobre a chegada de pacotes, mudanças de estados ou sobre algum erro. As quatro principais mensagens são:

I. **Packet-in:** transferir o controle de um pacote para o controlador.
II. **Flow-remove:** informar o controlador sobre a remoção de uma entrada de fluxo da tabela.
III. **Port Status:** informar status relativo às portas do *switch*.
IV. **Error:** notificar problemas ao controlador.

Por fim, as mensagens simétricas são enviadas sem solicitação, em qualquer direção, ou seja, o *switch* pode enviar esse tipo de mensagem para o controlador, como também o controlador pode enviar esse tipo de mensagem para o *switch*. São três tipos:

I. **Hello:** mensagens desse tipo são trocadas durante a estabilização da conexão.
II. **Echo:** responsável por verificar se a conexão entre ambos está ok, como também para medir a latência e largura de banda.
III. **Experimenter:** permite aos fabricantes implementarem funcionalidades adicionais utilizando este protocolo.

2.3 Considerações

O paradigma SDN é uma arquitetura recente, pois mesmo já existindo diversos equipamentos que abordam essa tecnologia, este modelo de rede guarda muitas incertezas em torno do seu funcionamento. Contudo, devido aos grandes avanços, principalmente do protocolo OpenFlow, há um grande interesse tanto acadêmico quanto empresarial em consequência das possibilidades que essa nova tecnologia abre para o futuro das redes de computadores. Dessa forma, este novo paradigma irá habilitar uma evolução significativa do

modelo tradicional de redes, permitindo aos administradores um gerenciamento direto e consolidado.

3 SISTEMA OPERACIONAL DE REDE – CONTROLADORES

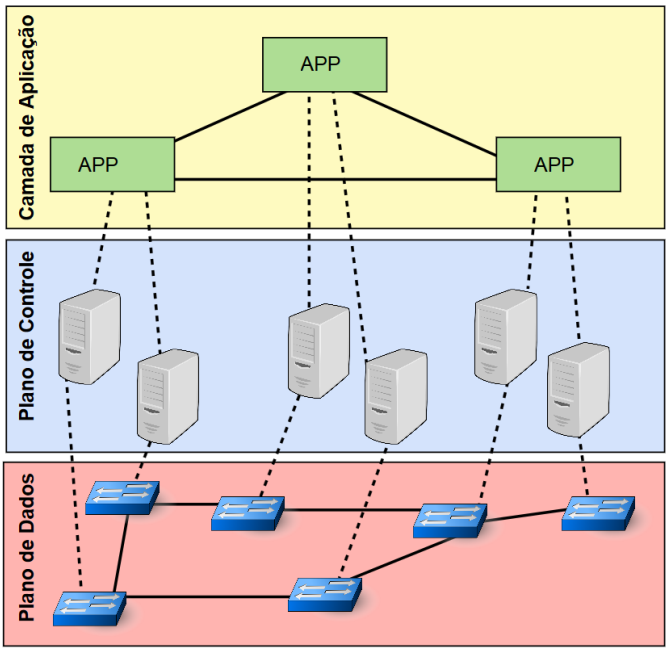
Neste capítulo têm-se como objetivo principal apresentar conceitos referentes aos sistemas operacionais de rede, denominados de controladores. Desse modo, o capítulo está organizado da seguinte forma: na Seção 3.1 são abordados os modos de organização do Plano de Controle de uma SDN, explicando principalmente as diferenças existentes nas três formas de organização, em seguida, na Seção 3.2, serão analisados aspectos referentes aos controladores, abordando seus conceitos e modo de funcionamento e, por fim, na Seção 3.3, são apresentadas algumas considerações finais referentes aos controladores SDN.

3.1 Organização do Plano de Controle

O plano de controle estabelece um conjunto de regras utilizadas para criar as entradas das tabelas de encaminhamento, sendo essas, por sua vez, consumidas pelo plano de dados para encaminhar o tráfego entre as portas de entrada e saída de um dispositivo de encaminhamento. Assim, o conjunto de informações referentes a topologia da rede, utilizadas para estabelecer as regras, são armazenadas na RIB (*Routing Information Base*). Dessa forma, para que a RIB se mantenha consistente, ocorre a troca de informações entre as diferentes instâncias do controlador presente na rede. Uma vez que a RIB é considerada consistente e estável, ocorre a programação da FIB (*Forwarding Information Base*), que nada mais é do que tabelas de entradas de encaminhamentos [NADEAU; GRAY, 2013; FARIAS; DINIZ; LUCENA, 2014].

A mecânica de organização do plano de controle e do plano de dados é demonstrado na Figura 5, que representa um padrão de rede SDN com *switches* interconectados. Desse modo, o plano de controle entende como funciona a topologia da rede e toma as decisões através de um ou mais controladores sobre o fluxo do tráfego, mantendo assim o controle sobre o plano de dados. Conforme afirma Kreutz *et al.*, [2014], o plano de controle pode ser representado por todos os protocolos utilizados para que os dispositivos da rede entendam como devem encaminhar o fluxo de dados.

Figura 5 - Organização do plano de controle e do plano de dados.
Fonte: Adaptado de Kreutz *et al.*, [2014].



O protocolo OpenFlow determina em suas especificações que os dispositivos de encaminhamento são gerenciados por controladores, implicando assim em uma centralização, ou seja, em um único ponto de falha. Dessa forma, diversos estudos foram feitos em torno desse modelo e foram definidas três arquiteturas para o plano de controle que serão descritas nas subseções a seguir. São elas: (i) plano de controle clássico centralizado, modelo tradicional que implica em um único ponto de falha, (ii) plano de controle totalmente distribuído, modelo mais robusto que conta com mais de um controlador, (iii) plano de controle distribuído hierárquico ou híbrido, modelo que utiliza características de ambos modelos citados anteriormente [NUNES *et al.*, 2014].

3.1.1 Plano de Controle Clássico Centralizado

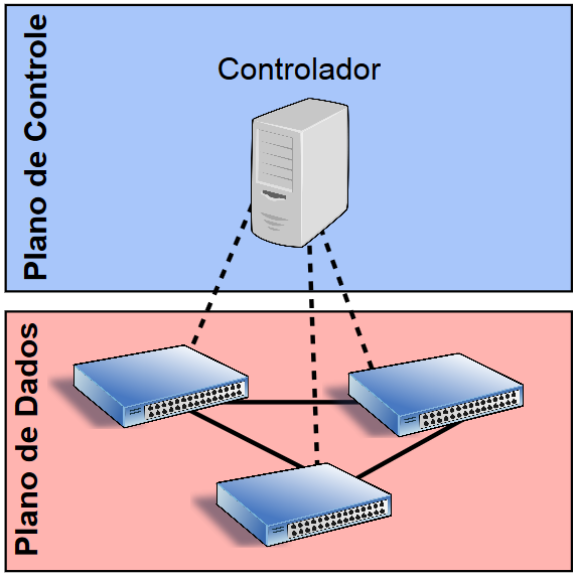
Um controlador centralizado fisicamente representa um único ponto de falha para a totalidade da rede, dessa forma, caso este controlador venha a falhar, a rede ficará inoperante. Entretanto, um controlador centralizado tem uma perspectiva mais ampla dos recursos que estão sob seu controle, e pode, potencialmente, fazer melhores decisões sobre o encaminhamento do fluxo de dados, além de garantir maior segurança, pois o fluxo de informações irá seguir apenas para esse controlador [CARIA; JUKAN; HOFFMANN, 2016].

Uma vez que o controlador logicamente centralizado tem uma visão global da topologia subjacente, é possível evitar, através disso, a disseminação de um mesmo pacote várias vezes sobre o mesmo link físico da rede. Entretanto, preservar esses benefícios em uma rede com o controlador centralizado é bastante desafiador. Por exemplo, em aplicações do tipo de

negociação financeira ou monitoramento de tráfego, não se analisa apenas a latência da rede, mas também o número de usuários que estão utilizando essa aplicação e como estes estão agindo. Assim, tem-se como consequência o gargalo em um único controlador, aumentando assim o tempo de resposta sobre os pedidos de atualização da rede que os dispositivos de encaminhamento enviam [BHOWMIK *et al.*, 2015].

Em um ambiente de rede com o controlador centralizado tem como desvantagem o fato de ele ser um ponto único de falha e possuir uma escalabilidade limitada. Entretanto, com a centralização do controlador é possível obter maior simplicidade na operação da rede e melhor visão sobre o seu comportamento, como pode-se perceber na Figura 6 um único controlador está conectado com todos os dispositivos de encaminhamento da rede. Dessa forma, fica claro que em ambientes em que se possui uma topologia de rede de menor tamanho e que o número de requisições para o controlador é reduzido, é válida esta implementação, pois assim estará garantindo maior simplicidade e uma melhor consolidação no gerenciamento da rede.

Figura 6 - Plano de Controle Centralizado.
Fonte: do autor.



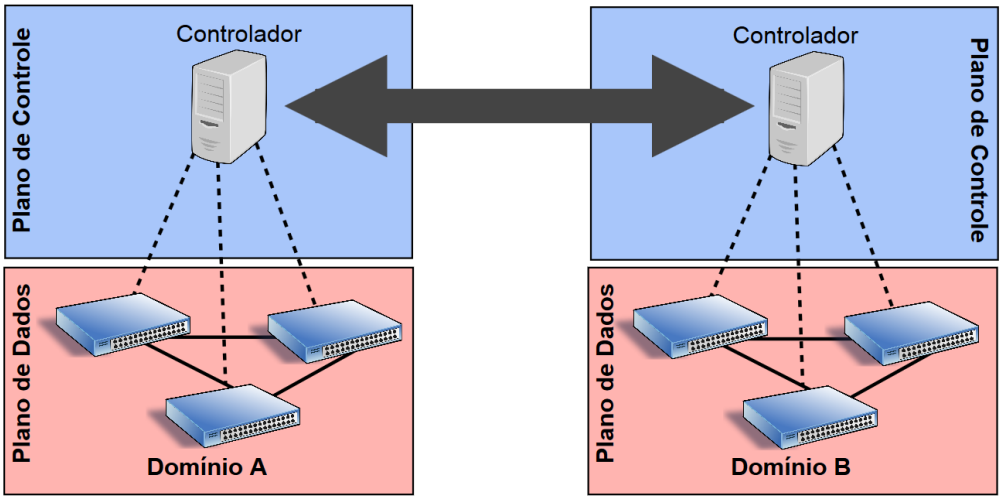
3.1.2 Plano de Controle Totalmente Distribuído

O Plano de Controle Distribuído surgiu com a ideia de solucionar as diversas desvantagens que o modelo centralizado traz. Dessa forma, trouxe consigo a concepção de um plano de controle consistente, tornando-se, portanto, um problema de computação distribuída, onde exige-se o mesmo raciocínio sobre as interações e simultaneidade entre os controladores, a fim de preservar o funcionamento correto do plano de dados. Assim, a simples replicação dos controladores não soluciona o problema, e sim cria outros, como por exemplo erros de sincronismo entre os controladores [CANINI *et al.*, 2015].

A principal técnica para aumentar resiliência e a escalabilidade de uma rede SDN é através da distribuição do controle da rede e da replicação dos controladores [MULLER *et al.*, 2014]. Entretanto, com a adoção do plano de controle distribuído os controladores irão agir como em um sistema distribuído, não sendo suficiente para garantir a segurança, um melhor desempenho e escalabilidade, pois além de uma correta implantação, necessita-se de um mapeamento otimizado entre os controladores e os dispositivos de encaminhamento, criando assim um domínio de controle [KREUTZ *et al.,* 2013]. Além disso, com o plano de controle distribuído, pode-se ter um domínio inconsistente da rede, acarretando em perda de desempenho e erros na execução de aplicações de controle. Assim, manter a visão global da rede centralizada através da distribuição do controle entre os diferentes nós físicos é um dos principais desafios do plano de controle distribuído, onde para alcançar essa solução é necessário determinar a correta localização e a quantidade necessária de controladores para uma determinada topologia de rede [HELLER *et al.,* 2012, BARI *et al.,* 2013].

Um único controlador pode garantir facilmente uma visão global de toda rede, garantindo assim consistência e melhores decisões sobre o encaminhamento do fluxo de dados [SCHIMID *et al.,* 2013]. No entanto, a situação muda quando o plano de controle se torna distribuído, porém, com um sistema distribuído, tem-se um grande número de vantagens, como por exemplo: maior escalabilidade, menor latência de um dispositivo de encaminhamento para seu controlador mais próximo, maior tolerância a falha e um balanceamento de carga, não existindo assim um gargalo na rede. Dessa forma, para uma correta administração do controle da rede é necessário separar os diversos *switches* e agrupá-los em domínios de controle distintos, onde cada controlador administra um conjunto separado de *switches* como ilustra a Figura 7.

Figura 7 - Plano de Controle Distribuído.
Fonte: do autor.



Neste contexto, os controladores podem ser organizados em um conjunto distribuído de controladores em diferentes domínios de rede. Assim, são exemplos de controladores distribuídos o Opendaylight [MEDVED *et al.*, 2014] e o ONOS [STANCU *et al.*, 2015]. Deste modo, para que estes controladores separados por domínios possam se comunicar e efetuar troca de dados ou verificar a consistência dos mesmos, é necessário que cada controlador implemente uma API denominada leste e oeste. Grande parte dos controladores distribuídos apresentam baixa consistência, isto é, as atualizações que ocorrem em nós distintos na rede são eventualmente atualizadas nas diferentes instâncias de controle presentes na rede. Em consequência disso, controladores distintos possuem visões diferentes para um mesmo conjunto de nós da rede. Uma consistência forte, garante que todos os controladores irão possuir uma mesma visão sobre os nós da rede, porém, mesmo sendo de simples implementação, não são todos os controladores que dispõem [KREUTZ *et al.*, 2014].

Para tanto, argumenta Levin *et al.*, [2012] que utilizando uma distribuição do controle físico, porém com o controle lógico centralizado, prejudica-se o desempenho da rede [LEVIN *et al.*, 2012]. Em contrapartida, para Schmid [2013], aplicações podem ser executadas por controladores centralizados, sem que haja necessariamente uma visão global consolidada da rede [SCHIMID; SUOMELA, 2013]. Com base nessas declarações, uma implementação de um plano de controle distribuído que seja tolerante a falhas, envolve diversos desafios que devem ser levados em conta. Provavelmente, o mais importante a se solucionar neste cenário, é a consistência fraca que os controladores possuem, não garantindo assim uma visão correta de todos os nós devido as atualizações que eventualmente acontecem.

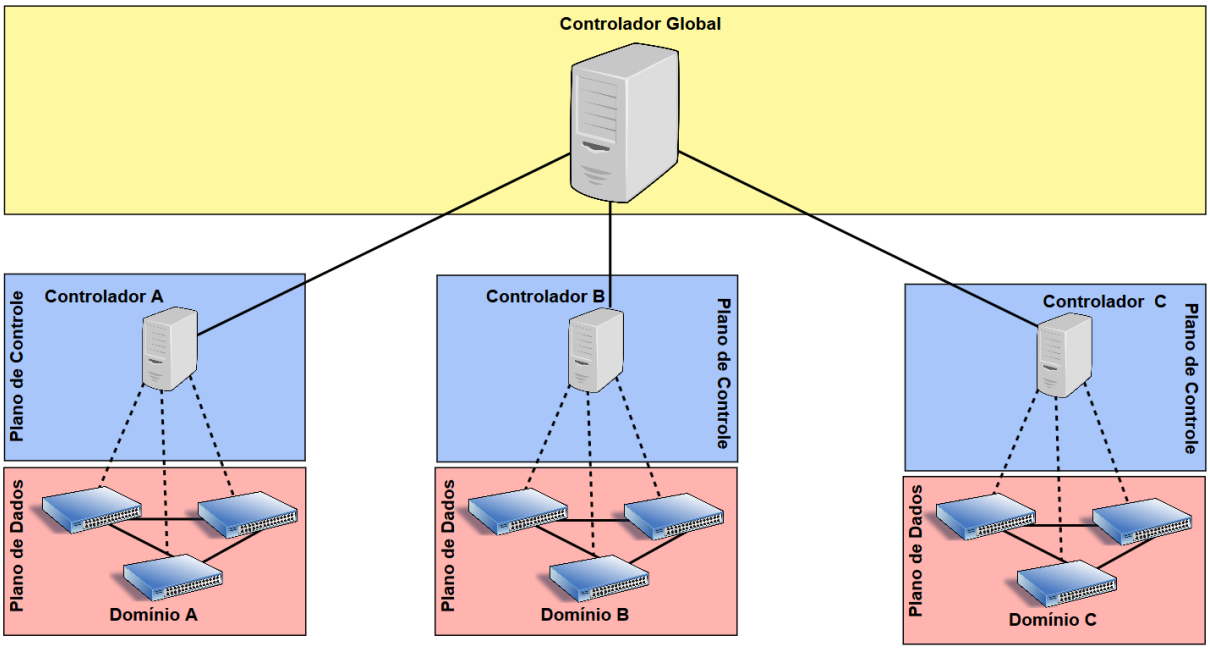
3.1.3 Plano de Controle Distribuído Hierárquico

A fim de aumentar a segurança e a resiliência na infraestrutura de comunicação das topologias SDN, foi desenvolvido um modelo de plano de controle novo, baseado nos sistemas distribuídos conhecidos, denominado de plano de controle hierárquico. Esse novo esquema de plano de controle fornece uma solução escalável para grandes áreas geográficas, baseando-se em características do plano de controle centralizado e do plano de controle distribuído, sendo assim um híbrido entre os dois [GENGE; PIROSKA, 2016; CARIA; JUKAN; HOFFMANN, 2016].

Para solucionar as limitações existente nos planos de controle centralizado e distribuído e aproveitar suas vantagens, um plano de controle híbrido faz-se necessário para alcançar essa coordenação. Deste modo, o plano de controle híbrido aproveita os benefícios de um simples controle de gestão de dados específico que existe no modelo centralizado e aproveita as vantagens da escalabilidade e resiliência que existe no modelo distribuído [HAKIRI *et al.*, 2014]. Além disso, o modelo de controle híbrido pode fornecer políticas de gestão capaz de resolver o problema de sincronização de estado, pois este modelo disponibiliza um controlador global que estará centralizado e irá coordenar os demais, a fim de que estes controladores sincronizem seus estados com base nesse coordenador global.

Uma abordagem híbrida, como Kandoo [YEGANEH *et al.*, 2012], pode utilizar controladores centralizados para aplicações locais e redirecionar para um controlador global as decisões que requerem sobre toda a rede. Isso reduz a carga no controlador global, pois filtra o número de novos fluxos. Dessa forma, os fluxos irão passar primeiro pelo controlador responsável pelo domínio e, quando a aplicação desejar um acesso ao estado de toda a rede, envia o fluxo de dados para o controlador global. O modelo Kandoo cria uma hierarquia de dois níveis para controladores: (i) controladores locais, responsáveis por executar aplicações de seu domínio, (ii) um controlador global, logicamente centralizado que controla os demais controladores. Tal como ilustrado na Figura 8, onde vários controladores locais são implantados ao longo da rede e são responsáveis por controlar um conjunto de *switches*.

Figura 8 - Plano de Controle Hierárquico.
Fonte: do autor.



O modelo de organização híbrida dos controladores SDN promete a possibilidade de uma migração da rede sem que ocorra interrupção ou inoperância, permitindo atualizações na infraestrutura existente sem a necessidade de mudar o sistema global. Dessa forma, um modelo híbrido aborda o crescimento da complexidade da rede, possibilitando assim maior confiabilidade e tolerância a falhas, sendo amplamente reconhecido na comunidade acadêmica.

3.2 Controladores

A camada de controle é responsável pela configuração dos elementos de encaminhamento da rede. Assim, o controlador SDN é uma entidade lógica centralizada que identifica as instruções necessárias para então configurar a infraestrutura da rede com base nos requisitos da camada de aplicação. Para gerenciar com eficiência a rede, os controladores SDN podem solicitar informações sobre a infraestrutura, tais como: estatísticas de fluxo, informações sobre a topologia e status sobre o funcionamento dos elementos de rede. Dessa forma, a entidade de software que implementa o controlador SDN é muitas vezes chamada de Sistema Operacional de Rede, devido ao seu modo de funcionamento [KREUTZ *et al.*, 2014; NUNES *et al.*, 2014].

O protocolo OpenFlow mesmo estando em um estágio avançado de desenvolvimento, não especificou como os controladores SDN devem ser implementados, dessa forma, um controlador pode ser implementado independentemente do modelo SDN existente, isto é, sem a necessidade de receber qualquer suporte. Por outro lado, um controlador além de gerenciar os dispositivos de rede, são capazes de fornecer recursos avançados, tais como virtualização, programação de aplicativos e até mesmo gerenciamento de banco de dados. Devido a esse

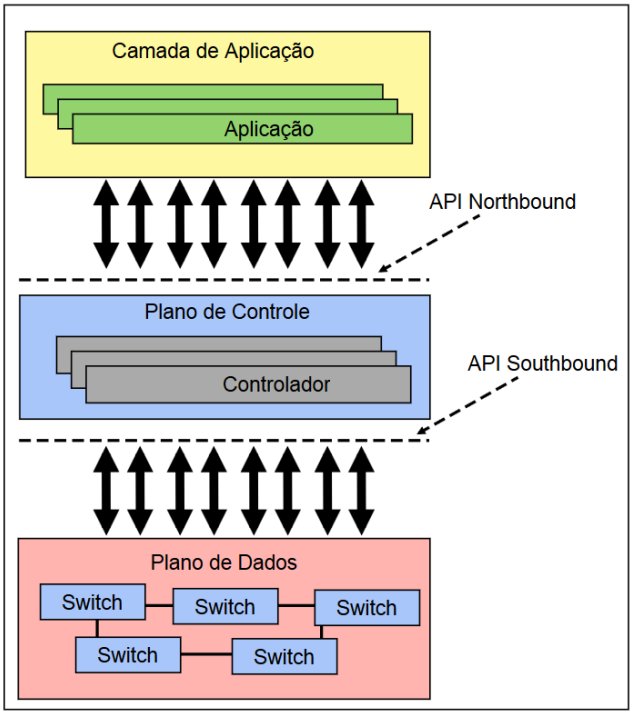
grande número de funcionalidades que um controlador SDN pode ter, é comum considerá-lo o cérebro de uma rede SDN [KREUTZ *et al.,* 2014].

O controlador foi desenvolvido para suprir a necessidade de um novo nível na arquitetura SDN, responsável então por concentrar as tarefas de gerenciamento dos dispositivos de encaminhamento e oferecer uma maior abstração para os desenvolvedores, definindo assim a natureza de uma SDN [KREUTZ *et al.,* 2014]. Dessa forma, o controlador é capaz de concentrar a comunicação com todos os elementos programáveis que estão compondo a rede e oferecer uma visão completa do seu estado [CASADO *et al.,* 2010]. Através dessa visão centralizada e global do estado da rede, é possível que seja desenvolvido análises detalhadas sobre a rede e, por meio desta, tomar decisões sobre como o sistema deverá agir.

Um controlador logicamente centralizado, não necessariamente determina que este controlador opera de maneira centralizada em um único ponto. Assim, é possível que se tenha uma visão global da rede com os controladores fisicamente distribuídos, seja através de uma divisão em domínios ou não.

Para que ocorra a comunicação entre a camada de aplicação e a camada de controle, e então sejam fornecidos os recursos necessários para o usuário final, faz-se necessário o uso de uma API denominada *Northbound*. Essa API é desenvolvida sobre o controlador e representa uma interface comunicação norte, onde normalmente abstrai um conjunto de instruções utilizadas pela interface sul para programar os dispositivos de encaminhamento. E no caso da comunicação entre a camada de controle e a camada de infraestrutura é utilizado outra API denominada *Southbound*. No caso dessa API, um conjunto de instruções para os dispositivos de encaminhamento são definidos, como também o protocolo de comunicação entre os dispositivos de encaminhamento e os elementos do plano de controle, formalizando assim a forma como os elementos de controle e os elementos do plano de dados irão interagir [NUNES *et al.,* 2014]. Essa arquitetura é ilustrada na Figura 9.

Figura 9 - Interfaces de Comunicação Northbound e Southbound.
Fonte: Adaptado de Kreutz *et al.*, [2014].



Diversos modelos de controladores já foram desenvolvidos dentro do contexto de SDN. Dessa forma, a linguagem de programação que cada um oferece pode ser considerada seu diferencial, determinando algumas vezes as funcionalidades que poderão ser oferecidas. Desse modo, nas subseções seguintes é realizada uma breve introdução dos três modelos de controladores escolhidos para uso no desenvolvimento desse trabalho de conclusão. São eles: (i) Floodlight, (ii) POX, (iii) Ryu. A escolha desse conjunto de plataformas de controle se deu baseado no levantamento da literatura relacionada. Foi identificado no estado-da-arte, que trabalhos com alto impacto na área recorreram a tais controladores para a prova de conceitos. Somado a popularidade destas plataformas estão também, a documentação da arquitetura disponível, acesso ao código fonte e APIs para prototipação de serviços de rede.

3.2.1 Floodlight

Floodlight é um controlador SDN de código aberto de classe empresarial licenciado pela Apache e desenvolvido em Java. Este controlador tem suporte para o protocolo OpenFlow da versão 1.0 a 1.4 e recebe suporte de uma comunidade de desenvolvedores, que o mantém sempre atualizado, oferecendo, além disso, uma arquitetura modular de fácil expansão, baseada em operações *multithread* [NETWORK, 2016]. Dentre seus objetivos estão a operação da plataforma com alto desempenho, além de servir como base para diversos outros projetos da sua empresa mantenedora, a *Big Switch Networks*. [FERNANDEZ, 2013].

3.2.2 POX

POX é uma plataforma de controle de código aberto desenvolvida em Python, com suporte a versão 1.0 do OpenFlow. Algumas vantagens deste controlador incluem uma interface intuitiva para o usuário e um desenvolvimento mais rápido e eficiente, por ser desenvolvido em uma linguagem de alto nível. Entretanto, ao contrário do Floodlight que trabalha com *multithread*, este controlador opera com apenas uma *thread*, o que acaba por limitar seu desempenho quando se aumenta o número de dispositivos na rede. Neste contexto, este controlador apresenta uma presença maior na área de pesquisa científica, não sendo muito utilizado na indústria. Por fim, o POX oferece suporte para diversos sistemas operacionais [OpenFlow, 2016; KAUR; SINGH; GHUMMAN, 2014].

3.2.3 Ryu

O controlador Ryu, assim como os anteriores, também é um controlador de código aberto e, como POX, também é desenvolvido em Python. Seu projeto tem como objetivo aumentar o desempenho da rede e tornar seu gerenciamento mais fácil. O Ryu fornece APIs bem definidas que torna mais fácil para os desenvolvedores criarem novas aplicações de gestão de rede e de controle. Esta plataforma adota o conceito de componentes que ajuda as organizações a customizarem as implementações para atender às suas necessidades específicas e, dessa forma, os desenvolvedores podem facilmente modificar componentes existentes ou implementar seu próprio componente para garantir que a rede poderá atender às novas demandas de suas aplicações [RYU Documentation, 2016].

3.3 Considerações

Como pode-se perceber neste capítulo, o plano de controle de uma SDN é muito complexo e deve ser levado a sério no momento de sua implantação. Assim, a escolha de um plano de controle centralizado, distribuído ou até mesmo híbrido, não pode ser discricionária. Deve ser levado em conta a topologia da rede e como os dispositivos de rede estão distribuídos geograficamente. Além disso, a escolha do sistema operacional de rede, ou controlador, é muito importante, pois ele será quem irá comandar toda rede.

Neste contexto, como já foi abordado, diversos controladores já foram desenvolvidos quando se fala em Redes Definidas por Software. Entretanto, muitos deles apresentam propósitos diferente. Assim, a escolha de um controlador correto e adequado a ser implantado em uma rede torna-se um desafio muito complexo a se enfrentar, pois deve ser levado em conta diversos quesitos que serão importantes após a implantação, como, por exemplo: garantias relacionadas a escalabilidade e à lógica de controle centralizada.

4 ESTADO DA ARTE EM BENCHMARK DE CONTROLADORES

Neste capítulo têm-se como objetivo principal apresentar informações referente ao estado da arte relacionado à *benchmark* de controladores. À vista disso, o capítulo está organizado da seguinte forma: na Seção 4.1 é apresentada uma visão geral referente à importância de um estudo precoce sobre o plano de controle, como também informar a respeito de como as ferramentas de *benchmark* auxiliam nesse estudo. Seguindo, na Seção 4.2, são analisadas implementações referentes a essas ferramentas de *benchmark* de controladores. Na Seção 4.3 apresenta-se uma síntese sobre as implementações de controladores. A Seção 4.4 apresenta uma comparação com o trabalho proposto e, por fim, a Seção 4.5 apresenta algumas considerações finais.

4.1 Visão Geral

O plano de controle é considerado uma parte essencial na arquitetura SDN, sendo assim muito importante receber uma atenção especial no momento da implantação. Devido a isso, diversos estudos têm sido feitos com o intuito de avaliar, testar e comparar o desempenho dos mais variados modelos de controladores, como também buscar respostas que facilitem a escolha do correto dispositivo de controle para determinado cenário de rede.

Ao contrário de *benchmarks* convencionais que se concentram em geral na taxa de transferência e/ou na latência, diversas ferramentas foram desenvolvidas com o propósito de abranger um número maior de quesitos em torno dos controladores OpenFlow [JARSCHEL *et al.*, 2012]. Nesta direção, Tootoonchian *et al.*, [2012] afirma que o desempenho de um controlador OpenFlow não está relacionado apenas à resultados obtidos através de *benchmarks* tradicionais, mas também em como o tráfego de controle é processado. Desse modo, a avaliação dos controladores torna-se uma etapa crucial que antecede o momento da implantação de uma rede SDN.

Neste contexto, no que se refere aos diversos controladores OpenFlow existentes, há questões sobre desempenho, robustez e confiabilidade que carecem de respostas para subsidiar a tomada de decisão em um projeto de migração para rede SDN. Entretanto, respostas desse tipo só são obtidas através de emulações de cenários e de topologias, onde é possível avaliar, através das ferramentas então existentes, e obter uma análise mais detalhada dos pontos críticos de cada controlador, bem como seu comportamento em cenários diferentes de rede [SEZER *et al.*, 2013].

4.2 Implementações

Nos últimos anos, diversos estudos foram realizados com um enfoque em redes SDN, dessa forma diversos avanços foram obtidos. Entretanto, questões sobre avaliação de desempenho em controladores OpenFlow é algo que deve ser levado a sério, principalmente quando precede a implantação de uma rede SDN. Neste contexto, diversas ferramentas foram desenvolvidas com este propósito, avaliar controladores OpenFlow, porém algumas apresentam resultados pouco significativos, principalmente por não emularem corretamente propriedades reais de rede [JARSCHEL *et al.,* 2012]. Nesta seção são apresentadas as implementações de maior popularidade na literatura referente a essas ferramentas que, posteriormente, serão utilizadas no decorrer desse trabalho de conclusão. São elas: (i) Cbench, (ii) OFCBenchmark, (iii) OFCProbe, (iv) HCProbe.

4.2.1 Cbench

Cbench é um software específico para realizar testes de desempenho com controladores OpenFlow. Esta ferramenta tem sido muito utilizada para medir o atraso que um controlador possui quando recebe mensagens do tipo *packet-in* de *switches* conectados a ele. Desse modo, Cbench emula um *switch* que se comunica com um controlador, sendo responsável assim por medir o desempenho deste e oferecer como resultados medições de latência para requisições enviadas do *switch*, como também informar o limite máximo de mensagens que o controlador pode suportar [TOOTOONCHIAN *et al.,* 2012].

Cbench, desde o seu desenvolvimento, tornou-se a ferramenta de avaliação padrão para o desempenho de controladores OpenFlow. No entanto, segundo Tootoonchian *et al.,* [2012], utilizar esta ferramenta para destacar melhorias no desempenho de controladores SDN é pouco válido, em termos de comportamento do controlador. Essa afirmação ocorre pelo fato da ferramenta ser *single-threaded*, ou seja, ser necessário que várias instâncias tenham sido iniciadas para utilizar múltiplas CPUs. Além disso, Cbench utiliza uma única conexão de controladores para todos os *switches* emulados por ele. Desse modo, as estatísticas geradas são recolhidas por todos os *switches* e não individualmente, sendo assim basicamente impossível avaliar o desempenho global de um controlador quando se possui um comportamento mais complexo no cenário de rede. Devido a tais limitações, a ferramenta Cbench é mais utilizada no âmbito acadêmico [JARSCHEL *et al.,* 2012].

4.2.2 OFCBenchmark

A ferramenta OFCBenchmark, assim como a ferramenta Cbench, consiste em um software com o propósito de avaliar o desempenho de controladores OpenFlow. A arquitetura

e a implementação dessa ferramenta foram definidas com base nos seguintes objetivos [JARSCHEL *et al.,* 2012]:

I. Escalabilidade: a ferramenta foi projetada de modo que várias instâncias possam ser executadas de forma coordenadas em diferentes núcleos de CPUs. Possibilitando assim que a carga gerada para avaliar um controlador não seja limitada por um único núcleo da CPU.

II. Estatísticas detalhadas de desempenho: foi desenvolvida para realizar medições de RTT, pacotes enviados e pacotes recebidos por segundo para cada *switch* conectado na rede. Desse modo, é possível manter atualizadas as estatísticas de cada *switch*, como também emular diferentes cenários de rede.

III. Modularidade: ao passo que o processo de desenvolvimento de controladores OpenFlow avança rapidamente, é necessário que a ferramenta seja adaptável a novos cenários, sendo possível, além disso, incluir novas métricas de desempenho e outros parâmetros para avaliar controladores.

Nesta direção, a arquitetura do OFCBenchmark consiste em três componentes principais*:* OFCBenchmark *Control Center* (OCC), OFCBenchmark *Client* (OC), e um *Virtual Switch* (VS). O OFCBenchmark *Control Center* consiste em uma interface gráfica para o usuário desenvolvida em Delphi, dessa forma, os parâmetros para avaliação devem ser configurados nessa interface de acordo com as necessidades. O componente OFCBenchmark *Client* possibilita a criação de uma topologia especificada em um arquivo de configuração. Por fim, o *Virtual Switch*, também considerado componente chave na arquitetura do OFCBenchmark, possui uma tabela de fluxo simplificada, capaz de responder às diversas solicitações do controlador. Além disso, cada *switch* virtual possui duas conexões de *socket* e três *threads* encapsuladas em cada objeto, possibilitando, através disso, tratar cada *switch* virtual como uma verdadeira entidade individual, fornecendo estatísticas detalhadas de desempenho [JARSCHEL *et al.*, 2012].

A ferramenta OFCBenchmark utiliza uma abordagem distribuída, ou seja, a avaliação de desempenho pode ser dividia entre os vários *hosts* presentes na rede. De modo que cada um desses *hosts* executem uma única instância do OFCBenchmark *Client* [JARSCHEL *et al.,* 2012]. Dessa forma, é possível realizar emulações de cenários reais de rede, possibilitando assim verificar o gargalo no controlador, como também inserir novos parâmetros de testes de acordo com a topologia.

4.2.3 OFCProbe

OFCProbe é uma extensão da ferramenta já analisada na seção anterior, OFCBenchmark. Essa nova ferramenta apresenta uma arquitetura modular que permite uma análise específica e profunda sobre o comportamento e características de um controlador OpenFlow. Desse modo, com uma visão mais detalhada e profunda é possível obter uma análise mais precisa do gargalo no desempenho do controlador, bem como analisar comportamentos inesperados. Assim, com base na ferramenta OFCBenchmark, OFCProbe traz consigo cinco objetivos principais [JARSCHEL *et al.*, 2014]:

I. Plataforma independente: para contornar o problema das limitações que certos sistemas operacionais possuem, a ferramenta OFCProbe foi desenvolvida com a ideia de ser executável nas arquiteturas de sistemas mais comuns.
II. Escalabilidade: capacidade de o software ser executado de forma coordenada em vários núcleos da CPU. Garantindo assim que a ferramenta não se limite por um único núcleo, assim como a ferramenta OFCBenchmark.
III. Modularidade: chave principal para ser capaz de se adaptar facilmente às novas versões dos controladores OpenFlow, sendo possível adicionar novos recursos e funcionalidades.
IV. Análise de desempenho: é o foco dessa ferramenta, onde se encontra a: análise de *throughput* e a análise de latência, ambas em situações de sobrecarga na rede, gravando o comportamento do controlador em diferentes cenários.
V. Estatísticas detalhadas: incluem um conjunto de recursos que permitem uma análise no comportamento dos controladores OpenFlow.

A arquitetura da ferramenta OFCProbe é independente e escalar, sendo assim possível realizar métricas sobre o comportamento de controladores independentemente da plataforma escolhida. Desse modo, OFCProbe emula *switches* virtuais, que são organizados em topologias, e que fornecem estatísticas granulares sobre diferentes aspectos do comportamento dos controladores [JARSCHEL *et al.*, 2014].

4.2.4 HCProbe

HCProbe é uma ferramenta similar às ferramentas de *benchmarks* convencionais. Essa ferramenta emula um número ilimitado de *switches* OpenFlow e *hosts* conectados a um controlador. Desse modo, utilizando HCProbe pode-se testar e analisar vários índices de operações do controlador de maneira simples. Além disso, HCProbe foi desenvolvido em Haskell, permitindo assim aos usuários criarem facilmente seus próprios cenários para o

controlador. Com base neste contexto, as principais características dessa ferramenta são [SHAMILOV *et al.*, 2013]:

I. Ser implementada em uma linguagem de alto nível, na qual facilita a extensão.
II. Apresenta API para testes personalizados.
III. Introduz uma linguagem específica de domínio incorporado para criar testes.

4.3 Síntese do Estado da Arte

Tabela 1 - Comparação das características das ferramentas analisadas.

Ferramenta	**Características**	**Arquitetura**	**Número de *Switches***
CBench	I. Avaliação de desempenho do controlador. II. Medição de latência para requisições enviadas do *switch* para o controlador. III. Estatísticas gerais relacionadas a todos *switches*.	*Single-Thread*	Capaz de emular apenas um *switch* conectado ao controlador.
OFCBenchmark	I. Avaliação de desempenho do controlador. II. Software modular. III. Parâmetros para avaliações configuráveis em uma interface gráfica. IV. Possibilidade de criar topologias específicas. V. Estatísticas relacionadas a cada *switch* na rede.	*Multithread*	Capaz de emular diferentes *switches* conectados ao controlador.
OFCProbe	I. Avaliação de desempenho de controladores. II. Parâmetros para avalições configuráveis em uma interface gráfica. III. Análise de *throughput* e latência. IV. Estatísticas detalhadas sobre cada controlador e sobre os switches conectados a ele.	*Multithread*	Capaz de emular diferentes *switches* conectados ao controlador.

HCProbe	I. Avaliação de desempenho e análises mais específicas de responsabilidade do usuário da ferramenta. II. Oferece uma linguagem específica para criação de testes. III. Permite estudar o comportamento de cada controlador através de um teste específico criado.	*Multithread*	Capaz de emular diferentes *switches* conectados ao controlador

Fonte: TOOTOONCHIAN et al., [2012]; JARSCHEL et al., [2012]; JARSCHEL et al., [2014]; SHAMILOV et al., [2013].

Com base na Tabela 1, pode-se visualizar uma comparação entre as quatro ferramentas de *benchmark* de controladores encontradas na literatura, sendo possível perceber que a avaliação comum a todas elas é a de desempenho de controladores. Entretanto, cada uma dessas ferramentas possui características e arquiteturas diferentes.

As ferramentas OFCBenchmark e OFCProbe, por exemplo, garantem maior conformidade com base em cenários reais, possibilitando criar cenários específicos para determinados tipos de testes. Entretanto, estas ferramentas não fornecem resultados referentes a quantidade máxima de mensagens que cada controlador é capaz de suportar. Dessa forma, caso a capacidade do controlador seja menor que a quantidade de fluxo de dados gerados pela rede, o controlador será sobrecarregado, podendo gerar latência na comunicação. Em contrapartida, a ferramenta CBench, por sua vez, fornece testes que possibilitam descobrir a quantidade máxima de mensagens suportada por cada controlador, porém essa ferramenta é capaz de simular apenas um *switch* conectado a cada controlador, o que torna inviável em cenários reais de rede. Desse modo, pode-se perceber que as ferramentas então existentes fornecem resultados iniciais relevantes, auxiliando nas análises referentes a rede e ao controlador. Contudo, por não seguirem uma metodologia adequada, apresentam limitações quanto as conclusões que podem ser obtidas para tomada de decisões no projeto de redes SDN.

4.4 Diferenças em Relação a Proposta

Em vista do que foi apresentado nas subseções anteriores, nota-se que existem poucas respostas que determinem padrões para avaliações de desempenho em controladores OpenFlow, implicando assim no desenvolvimento de um conjunto de ferramentas que fornecem resultados limitados. Dessa forma, neste trabalho de conclusão busca-se desempenhar avaliações de forma experimental com o intuito de identificar a influência de diversos fatores

(por exemplo, características topológicas, demanda, controle de tráfego e serviços específicos de rede) no desempenho dos controladores e, além disso, pretende-se utilizar como referências documentos fornecidos por Vengainathan *et al.,* [2016], documentos estes que definem uma metodologia para avaliações de desempenho em controladores. Essas metodologias por sua vez, são uma extensão da RFC 7426 já definida. Os termos fornecidos nestes documentos ajudam a definir métricas sobre avaliações de desempenho de controladores independentemente dos protocolos suportados e serviços de redes utilizados [VENGAINATHAN *et al.,* 2016].

4.5 Considerações

Dada a importância do controlador OpenFlow, visto que este fornece uma plataforma de operação para diversas aplicações de controle e gerenciamento, é fundamental compreender o desempenho e o comportamento deste software para a operação em uma SDN. Entretanto, pouco se sabe sobre a estabilidade e desempenho dessas plataformas de controle. Desse modo, escolher o controlador correto para um dado cenário de rede exige uma análise aprofundada sobre esses quesitos nos diversos modelos existentes. Em vista disso, as ferramentas apresentadas fazem parte do esforço inicial a fim de auxiliar na escolha do software de controle adequado.

5 METODOLOGIA DE AVALIAÇÃO DAS FERRAMENTAS DE BENCHMARK

Neste capítulo, têm-se como objetivo principal apresentar as metodologias adotadas para a análise das ferramentas de *benchmark* de controladores, sendo essas, parte da solução proposta especificada. Desse modo, o capítulo está organizado da seguinte forma: inicialmente, na Seção 5.1, é apresentada a escolha de duas ferramentas de *benchmark* de controladores. A seguir, na Seção 5.2, descreve-se o cenário de avaliação dessas ferramentas, nas Seção 5.3 e 5.4, é descrito o que será analisado em cada ferramenta. Ao final, na seção 5.5 é apresentado o software Collectl utilizado para análise dos recursos computacionais disponíveis da máquina hospedeira.

5.1 Escolha das Ferramentas de Benchmark

Frente ao conjunto de ferramentas de benchmark apresentado e discutido no Capítulo 4, foi preciso reduzir o escopo de avaliação para um subconjunto das ferramentas, dada a característica combinatória desse tipo de trabalho experimental. Para tanto, a escolha das ferramentas adotadas para esse trabalho seguiu as seguintes premissas: popularidade, relevância dos seus resultados e disponibilidade do código aberto para que fosse possível investigar as causas de diferentes situações que são provocadas em trabalhos de natureza experimental. Dessa forma, das ferramentas apresentadas, as duas que permaneceram após essa filtragem foram: CBench e OFCProbe. Além disso, a ferramenta CBench foi escolhida também pelo fato de ser a ferramenta padrão para todos os tipos de análises relacionadas a desempenho de controladores. Já, a ferramenta OFCProbe, desenvolvida em Java, tida como uma evolução da ferramenta OFCBenchmark, além dos fatos apontados, apresenta estatísticas detalhadas dos controladores avaliados como também o suporte a múltiplos *switches* conectados ao controlador.

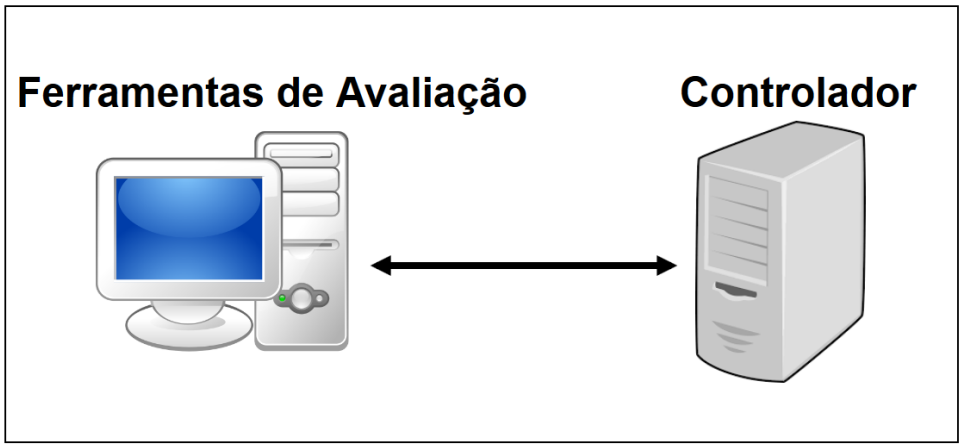
Para tanto, as análises dessas ferramentas de *benchmark* consistiram em uma série de experimentos nos quais foram avaliados fatores referentes aos controladores como também relacionados ao consumo de recursos da máquina hospedeira em que o controlador está instalado, para isso, foi utilizado um software específico para esse tipo de análise, denominado de Collectl. Por fim, vale lembrar que, os controladores utilizados para a análise foram descritos anteriormente nesse documento (na Seção 3.2): Floodlight, POX, RYU.

5.2 Cenário de Avaliação das Ferramentas de Benchmark

O cenário de avaliação definido para os experimentos das ferramentas de *benchmark* exige ambiente controlado, assim conta com duas máquinas conectadas em uma rede local, de modo que não ocorra nenhuma interferência da rede externa no tráfego de dados. Dessa forma,

em uma das máquinas são instaladas as duas ferramentas de avaliação, ferramentas estas que irão gerar os tráfegos para os controladores e então analisá-los. Em seguida, na segunda máquina tem-se a instalação dos três controladores que serão analisados em conjunto com o software Collectl, responsável por analisar os recursos da máquina hospedeira. Desse modo, ao iniciar as avaliações, apenas o controlador em questão a ser analisado é executado. Na Figura 10 tem-se o cenário desenvolvido para os experimentos.

Figura 10 - Cenário de Avaliação das Ferramentas de Benchmark
Fonte: do autor.



Como cada uma dessas ferramentas emula *switches* virtuais a serem conectados ao controlador, não foi preciso software de emulação adicional. Logo, para cada uma das ferramentas foram definidos parâmetros de configurações de acordo com o que cada um disponibiliza. Além disso, foi utilizado o software Collectl que é instalado na máquina hospedeira do controlador, ficando assim responsável por coletar informações referente à memória, disco, rede, paginação, entre outras propriedades.

Para a ferramenta CBench foram realizados dois tipos de experimentos (latência e vazão), os quais, para cada conjunto, foram realizadas três avaliações com parâmetros diferentes. Na primeira avaliação foi definida a seguinte parametrização: 8 *switches* virtuais emulados conectados a 100 hosts. Dessa forma, cada avaliação teve um total de 10 iterações com intervalos de 6 minutos entre cada uma delas, totalizando um total de 60 minutos por avaliação. Na segunda e terceira avaliação aumentou-se o número de *switches* para 16 e 32 respectivamente, mantendo-se os outros parâmetros. As configurações da ferramenta Cbench podem ser vistas na Tabela 2.

Tabela 2 - Parâmtros de Configuração da Ferramenta Cbench

	OpenFlow	*Switches*	Hosts	Intervalo (seg)	Iterações	Tempo (min)
1ª	1.3	8	100	360	10	60
2ª	1.3	16	100	360	10	60
3ª	1.3	32	100	360	10	60

A ferramenta OFCProbe, ao contrário da ferramenta CBench, fornece um grande conjunto de estatísticas por controlador e switches. Dessa forma, foram escolhidas as seguintes estatísticas a serem analisadas: pacotes enviados por segundos por *switches* e latência entre o controlador e os *switches*. Essas estatísticas foram escolhidas pelo fato de serem as únicas que se referem, em uma mesma avaliação, ao controlador e ao *switch*. Optou-se por utilizar a configuração padrão da ferramenta, alterando apenas três parâmetros: o número de *switches* por experimento, o número de hosts conectados por *switch* e o tempo total dos experimentos. A configuração da ferramenta OFCProbe pode ser visualizada na Tabela 3.

Tabela 3 - Parâmetros de Configuração da Ferramenta OFCProbe

	OpenFlow	Switches	Hosts	Tabela de fluxo	Threads por switch	Tempo (min)
1ª	1.3	8	100	128	4	60
2ª	1.3	16	100	128	4	60
3ª	1.3	32	100	128	4	60

5.3 Ferramenta CBench

Para a ferramenta Cbench, foram utilizados seus dois modos de operações: latência e vazão. Em modo latência, cada *switch* emulado mantém exatamente um único fluxo de dados pendente, o qual fica aguardando uma resposta antes de solicitar o próximo fluxo. Assim, o modo latência mede o tempo de processamento da solicitação do controlador em condições de baixa capacidade. Já, em contrapartida, no modo vazão cada *switch* mantém tantas solicitações pendentes quanto o armazenamento em buffer permitir, medindo assim, a taxa máxima de fluxo que o controlador pode suportar. Dessa forma, foram feitas diversas análises utilizando esses dois modos de operações, os quais, para cada um, foram realizados três experimentos com os diferentes números de *switches* parametrizados.

5.4 Ferramenta OFCProbe

A ferramenta OFCProbe, ao contrário da ferramenta Cbench, fornece em um único experimento diversas estatísticas. Assim, ao iniciar uma avaliação, é possível parametrizar

quais resultados que se deseja. Dessa forma, ao ser finalizado, a ferramenta gera diversos arquivos de dados, possibilitando uma análise posterior. Para os experimentos dessa ferramenta, foram escolhidas duas métricas: latência entre o controlador e os *switches*, o qual retorna resultados de testes de pings, ou seja, *Round Trip Time* (RTT), e a quantidade de pacotes enviados por segundos dos *switches* para o controlador, sendo possível analisar além disso, a quantidade de pacotes recebidos pelo controlador. Assim, foi padronizado a utilização dos mesmos processos para ambas ferramentas.

5.5 Software Collectl

O software Collectl não é uma ferramenta específica para avaliações de controladores, mas sim utilizado para analisar os recursos de uma máquina, neste caso, a máquina em que o controlador está instalado. Para tanto, foi necessário utilizar a ferramenta Cbench para gerar um grande número de fluxos de dados contra o controlador para que então o software Collectl pudesse analisar o comportamento da máquina hospedeira. Assim, para que o Collectl analisasse apenas informações referente ao controlador e ignorasse os outros processos que estavam rodando na máquina, foi preciso indicar, através de um arquivo de configuração, qual processo que ele deveria analisar, dessa forma, para cada controlador foi preciso alterar esse arquivo.

Para essa análise, teve-se como objetivo inicial gerar uma grande carga sobre o controlador, de modo que não o deixasse inoperante. Portanto, foi utilizada a seguinte parametrização na ferramenta Cbench: 8 *switches* virtuais, 1000 hosts conectados aos *switches* e 60 iterações com intervalo de 60 segundos entre cada uma. Dessa forma, foram analisadas as seguintes informações: consumo de memória RAM e uso da CPU.

6 PROTÓTIPO DA FERRAMENTA DE BENCHMARK

Este capítulo, têm-se como objetivo principal apresentar as informações referente ao desenvolvimento do protótipo da ferramenta de *benchmark* que compõe a parte final da solução proposta definida. Dessa forma, o protótipo desenvolvido seguiu com base em documentos recentes disponibilizados pela entidade IETF, conforme discussão realizada no Capítulo 4, Seção 4.6. A IETF possui vários grupos de trabalho que se concentram na construção de estruturas para medição de sistemas de rede, protocolos e padronizações no âmbito da área de Redes de Computadores. Em se tratando de controladores SDN, a IETF apresenta um projeto denominado de Metodologia de *Benchmarking* para Desempenho de Controladores SDN. Este projeto em questão, trabalha dentro de um outro grande grupo, denominado de Grupo de Trabalho para Metodologias de *Benchmarking* [VENGAINATHAN *et al.*, 2016].

Nesta direção, o protótipo desenvolvido foi baseado em duas métricas utilizadas para *benchmarking* de controladores definidas por essa entidade, são elas: medições de desempenho e análises de segurança. Desse modo, o capítulo está subdividido da seguinte forma: na Seção 6.1 são descritas informações referentes às métricas para avaliações de desempenho, na Seção 6.2, são apresentadas as análises referente à segurança. Ao final, na Seção 6.3 é apresentado o protótipo desenvolvido.

6.1 Desempenho

Para fins de avaliações, foi escolhido um aspecto considerado impactante em uma SDN: o tempo de descoberta da topologia da rede. Dessa forma, com base nos padrões definidos por Vengainathan et al., [2016], o tempo de descoberta da topologia da rede é a chave para que o controlador efetue seu gerenciamento. Portanto, se deve avaliar o tempo gasto pelo controlador para determinar a topologia da rede completa, no qual este tempo será definido como o intervalo entre o início da primeira mensagem de descoberta do controlador em sua interface Southbound e terminando com todas as características da rede determinada. Dessa forma, com base nos padrões definidos por Vengainathan *et al.*, [2016], são descritas a seguir as metodologias utilizadas para avaliar esse aspecto.

I. Verificar se o controlador está operacional
II. Estabelecer as conexões de rede entre o controlador e os dispositivos de rede.
III. Registrar o tempo para a primeira mensagem de descoberta recebida do controlador.
IV. Consultar o controlador a cada 3 segundos para obter informações da topologia da rede e compará-la com a rede implantada.

V. Parar o teste quando as informações da topologia descoberta corresponder à topologia da rede implantada, ou quando as informações das topologias descobertas para as 3 consultas retornarem o mesmo resultado.
VI. Registrar o tempo da última mensagem de descoberta enviada ao controlador.

6.2 Segurança

Para as análises de segurança, foi escolhido um aspecto considerado muito importante em uma SDN, seja ela de larga ou pequena escala: tratamento para negação de serviço. Este aspecto é responsável por determinar a consequência do tratamento de ataques do tipo DDoS (Distributed *Denial of Service*) em testes de desempenho. Assim, para que esse experimento possa ser realizado será executado antes um teste de latência na descoberta da topologia da rede. A seguir, a descrição da metodologia adotada neste aspecto[VENGAINATHAN *et al.*, 2016].

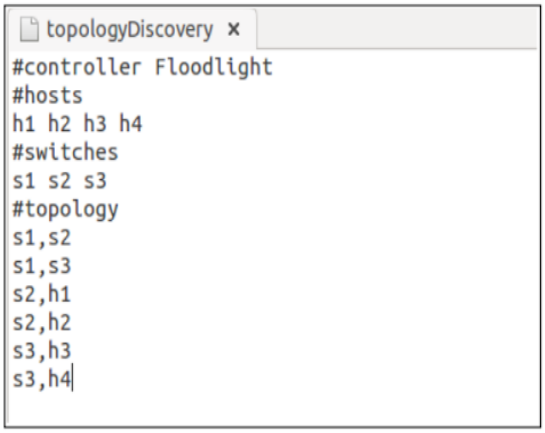
I. Executar o teste de latência na descoberta da topologia da rede.
II. Iniciar um ataque DDoS contra o controlador enquanto o teste de desempenho estiver em executação. Os ataques DDoS foram feitos através de um script criado em Python para gerar diversas solicitações de ping contra o controlador

6.3 Protótipo Desenvolvido

O protótipo da ferramenta desenvolvida foi feito utilizando a linguagem de programação C/C++. Além disso, em vista dos diversos modelos de controladores existentes, foram escolhidos dois controladores: Floodlight, pois dentre os controladores analisados é o que possui um maior desempenho por ser o único que suporta multithread, e o controlador POX em vista da sua popularidade e por ser muito utilizado no âmbito acadêmico em testes de SDN.

Pelo fato de ambos os controladores terem sido desenvolvido em linguagens de programação diferentes, Floodlight em Java e POX em Python, foi necessário implementar dois métodos separados que ficam responsáveis por descobrir a topologia da rede e direcionar a saída para um arquivo de texto sempre que recebem tal solicitação. Na Figura 11 tem-se o modelo de arquivo gerado ao descobrir a topologia da rede.

Figura 11 - Arquivo de Saída da Topologia Descoberta
Fonte: do autor.



Nesta direção, foram necessárias duas instâncias do protótipo desenvolvido, as quais foram denominadas de cliente e servidor, como ilustra a Figura 12. Desse modo, o protótipo cliente é responsável por toda a configuração da ferramenta, sendo esta feita em um arquivo de texto que segue o seguinte formato:

I. **controller**: campo onde deve ser informado qual controlador está sendo utilizado, “f” Floodlight e “p” POX.

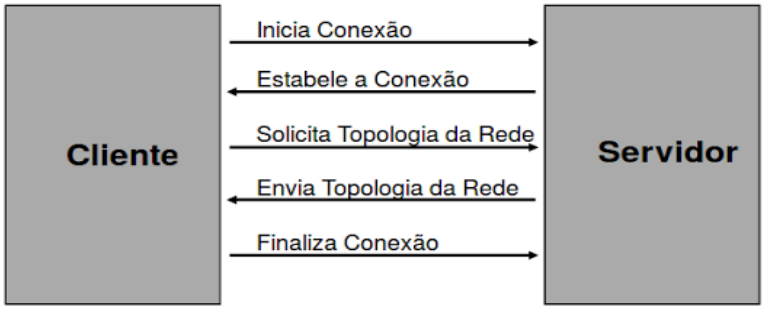
II. **IP**: campo responsável por informar o endereço IP da máquina em que o servidor está instalado.

III. **port**: campo responsável por informar a porta de comunicação utilizada pelo servidor.

IV. **mode**: campo referente ao modo de experimento que será executado, “s” segurança e “d” desempenho.

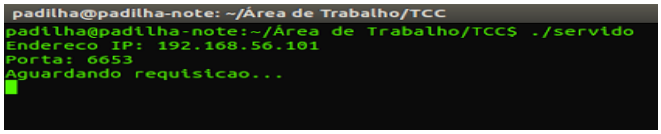
V. **format**: campo referente ao formado que a informação deverá ser tratada, isto é, milissegundos (ms), segundos (seg), minutos (min), horas (h).

Figura 12 - Modelo de Comunicação Entre o Cliente e o Servidor
Fonte: do autor.



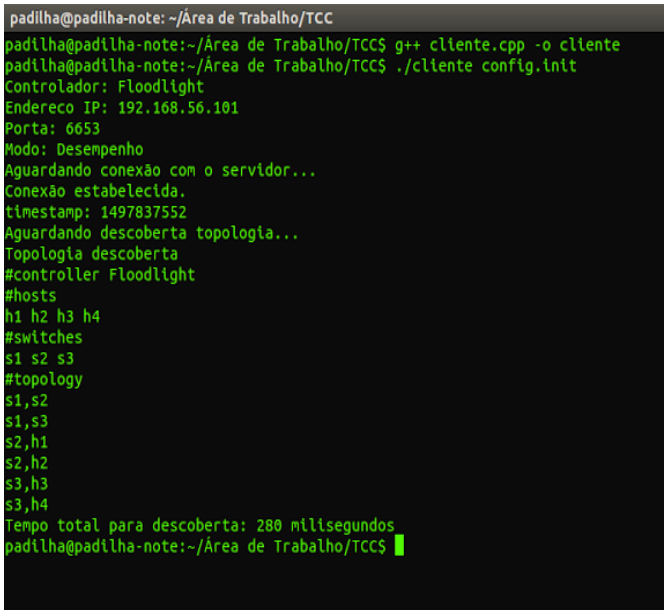
Todas as informações descritas no arquivo de configuração são utilizadas para a comunicação com o protótipo servidor, onde este por sua vez é responsável por estabelecer a comunicação com o controlador e solicitar a topologia da rede e então analisar as informações que o controlador fornece. Dessa forma, antes de iniciar o protótipo cliente, deve-se iniciar o servidor, para que este fique aguardando as requisições. Na figura 13 tem-se a imagem do servidor ao ser iniciado.

Figura 13 - Servidor Iniciado.
Fonte: do autor.



Para iniciar o cliente, é necessário indicar o caminho do arquivo de configuração por parâmetro. Dessa forma, o cliente irá enviar uma mensagem para o servidor com as configurações informadas no arquivo. Assim, o servidor irá se comunicar com o controlador, através do método criado, solicitando a topologia da rede. Após solicitar a topologia da rede, o servidor irá aguardar que o controlador escreva toda a topologia da rede no arquivo de texto, e então irá ler esse arquivo e retornar para o cliente. Na Figura 14 tem-se a imagem da execução do protótipo cliente, onde é utilizado o controlador Floodlight.

Figura 14 - Cliente Após a Execução
Fonte: do autor.



Conforme ilustra a Figura 14, foi utilizado o modo desempenho, no qual é avaliado o tempo que o controlador leva para descobrir todas as informações da rede e então retorná-la para o cliente. Assim, através dessas análises, foram feitos experimentos utilizando dois cenários com alterações na topologia da rede, os quais serão descritos no capítulo seguinte.

7 METODOLOGIA DE AVALIAÇÃO DO PROTÓTIPO DE BENCHMARK

Neste capítulo, têm-se como objetivo principal apresentar informações referentes a metodologia adotada para as avaliações do protótipo da ferramenta desenvolvida. Dessa forma, o capítulo está organizado da seguinte forma: primeiro, na Seção 7.1, é apresentado o ambiente onde serão realizados os experimentos da ferramenta desenvolvida. Na Seção 7.2 são descritas as especificações referentes ao ambiente de avaliação. Ao final, a Seção 7.3 apresenta o cenário onde as avaliações serão feitas, além dos aspectos que serão analisados.

7.1 Ambiente de Avaliação

Com base na literatura, grande parte dos trabalhos relacionados à SDN são desenvolvidos em ambientes onde é possível efetuar a emulação de cenários de rede. Dessa forma, para que os experimentos utilizando a ferramenta desenvolvida tenha êxito, foram escolhidas duas ferramentas que facilitam a prototipação e a execução de experimentos sem a necessidade da aquisição de equipamentos reais, ferramentas estas que serão descritas nas subseções seguintes. Nesta direção, foram escolhidas para o desenvolvimento deste trabalho de conclusão: (i) Mininet [LANTZ; HELLER; MCKEOWN, 2010], comumente utilizado para emulação de cenários em redes SDN, (ii) Open vSwitch [PFAFF *et al.*, 2015], projetado para ser um *switch* virtual rico em recursos também amplamente adotado na indústria. Assim, se construiu um testbed controlado de modo a viabilizar a execução da ferramenta de *benchmark* que foi desenvolvida. Neste testbed, através do software de emulação Mininet, os dispositivos de encaminhamento e os *hosts* são instanciados e gerenciados. E, neste testbed, os dispositivos de encaminhamento são *switches* virtuais Open vSwitch. No que segue são apresentados mais detalhes sobre as duas principais plataformas base do ambiente de experimentação – Mininet e Open vSwitch.

7.1.1 Mininet

Mininet é um sistema para prototipagem rápida de grandes redes, desenvolvido a fim de facilitar a avaliação de novos protocolos e aplicações SDN. Seu funcionamento se dá através de um ambiente emulado, sendo assim capaz de simular nós de uma rede utilizando recursos presentes no sistema operacional Linux. Dessa forma, Mininet foi projetado com base nos seguintes atributos [LANTZ; HELLER; MCKEOWN, 2010]:

I. Flexibilidade: novas topologias e novas funcionalidades devem ser definidas em software
II. Implementável: implementações de protótipos em redes baseada em hardware não devem exigir alterações no código ou na configuração do Mininet.
III. Interativo: o gerenciamento e a execução da rede devem ocorrer em tempo real.

IV. Escalável: o ambiente de prototipagem deve ser escalável para redes com centenas ou milhares de *switches*.

V. Realista: o comportamento do protótipo desenvolvido deve representar um comportamento real, com um alto grau de confiança.

VI. Compartilhável: protótipos desenvolvidos devem ser facilmente compartilhados com colaboradores, podendo assim executar e modificar de acordo com suas necessidades.

Atualmente, os ambientes de prototipagem disponíveis possuem seus prós e contras. À primeira vista, uma rede de máquinas virtuais pode ser atraente, no entanto a utilização de VMs podem gerar uma sobrecarga de memória, limitando assim a escalabilidades dos *switches* e *hosts*. Mininet por sua vez utiliza um conceito de virtualização leve, no qual é empregado a virtualização a nível de processos, onde muitos recursos do sistema são compartilhados. Dessa forma, usuários podem implementar novos recursos de rede ou arquiteturas inteiramente novas, testá-los em topologias emuladas e em seguida utilizar este mesmo código para testar em redes de produção [LANTZ; HELLER; MCKEOWN, 2010].

Assim como outras ferramentas existentes, na mesma linha, o Mininet também apresenta algumas limitações, as quais foram importantes tomar conhecimento antecipado para não agir de modo injusto. Dentre as limitações, aquela que foi levada em consideração para esse trabalho de conclusão foi a falta de confiabilidade no desempenho de suas aplicações em cargas elevadas. Os recursos da CPU são multiplexados em tempo real pelo escalonador padrão do Linux, dessa forma não se têm garantia de que um *host* que esteja pronto para enviar determinado pacote será escalonado imediatamente. Entretanto, mesmo com algumas limitações, Mininet funciona surpreendentemente bem, aproveitando recursos do Linux para implementar redes gigabits e centenas de nós (*switches*, *hosts* e controladores), sendo assim amplamente utilizado na literatura científica da área [LANTZ; HELLER; MCKEOWN, 2010].

7.1.2 Open vSwitch

Com o aumento da virtualização nos últimos anos mudou-se a maneira de como enxergamos a computação, essa mudança trouxe consigo a necessidade do surgimento de um novo elemento denominado de *switch* virtual. Na virtualização em rede, o *switch* virtual torna-se o principal provedor de serviços de rede, deixando a rede física com o transporte de pacotes encapsulados entre o *hypervisor* (camada de software entre o hardware e o sistema operacional responsável por monitorar a VM). Essa abordagem permite que as redes virtuais sejam desacopladas de suas redes físicas subjacentes, tornando assim, os *switches* virtuais, parte

integrante de qualquer configuração de computação virtualizada [PFAFF *et al.*, 2015; EMMERICH *et al.*, 2014].

Neste contexto, com o aumento da complexidade das redes virtuais e devido às limitações nos *switches* virtuais existentes, foi desenvolvida a ferramenta Open vSwitch. Diferentemente dos equipamentos de redes tradicionais, sejam eles software ou hardware, que alcançam alto desempenho através de suas especificações, o Open vSwitch é projetado para ser flexível e de uso geral. Ele deve alcançar alto desempenho sem a necessidade de especificações, adaptando-se às diferentes plataformas, enquanto compartilha recursos com o *hypervisor* [PFAFF *et al.*, 2015].

O Open vSwitch é comumente utilizado como *switch* em SDN e, a principal maneira de controlar o encaminhamento de pacotes é através do protocolo OpenFlow. Assim, para que ocorra o encaminhamento de pacotes são necessários dois componentes principais: ovs-vswitchd, um *daemon* responsável por controlar e gerenciar os *switches* virtuais; e um módulo *kernel* de *datapath*. Dessa forma, o ovs-vswitchd recebe uma tabela de fluxo OpenFlow a partir do controlador, compara essa tabela com quaisquer dos pacotes recebidos do módulo *datapath* e então reúne as ações aplicadas para armazenar o resultado no *kernel datapath*. Isso permite que o módulo do *datapath* permaneça sem conhecimento das particularidades do protocolo OpenFlow, tornando sua arquitetura mais simples. Além disso, por possuir compatibilidade com o protocolo OpenFlow e permitir que o consumo de memória seja reduzido através da utilização do modo *kernel* para encaminhamento, este software é muito utilizado para desenvolver cenários de avaliações de desempenho em SDN [PFAFF *et al.*, 2015; MARQUES, 2012].

7.2 Especificações do Ambiente

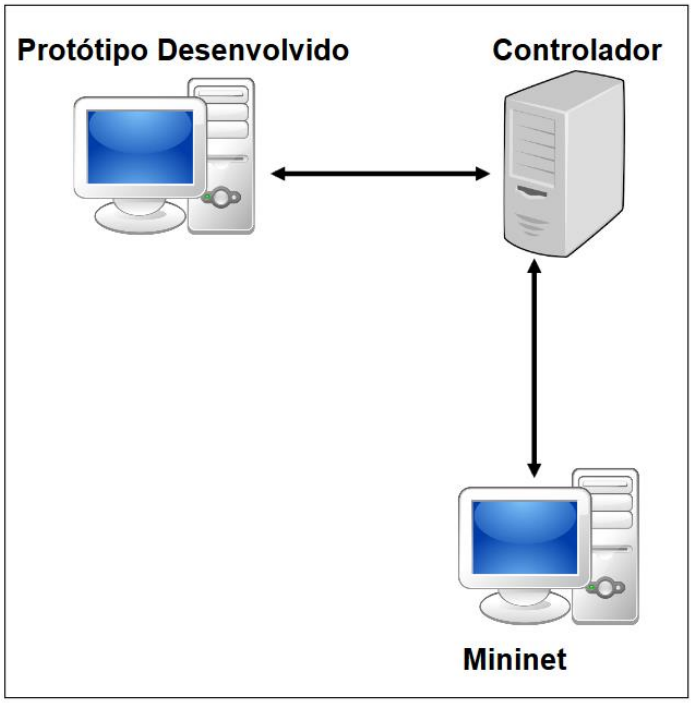
As avaliações da ferramenta desenvolvida foram executadas em uma máquina com processador Intel (R) Core (TM) i5-3210M CPU @ 2.50GHz, executando o sistema operacional Ubuntu (64 bits) 14.04.2 (kernel 3.16.0-37-generic x86-64) com 6 GB de memória RAM. O *hypervisor* da máquina executará o VirtualBox 5.1 para gerenciar a VM com o sistema operacional Ubuntu (64 bits) 14.04.2 (kernel 3.16.0-37-generic x86-64). Dentro das VMs, será utilizado o Open vSwitch 2.5.1 LTS e o Mininet 2.3.0d1.

7.3 Cenário de Avaliação

O cenário de avaliação, segue em linha com o ambiente elaborado para a avaliação das outras duas ferramentas de benchmark, conforme já descrito na Seção 5.2 (Figura 10). Para os experimentos do protótipo desenvolvido, foram utilizadas três máquinas conectadas em uma

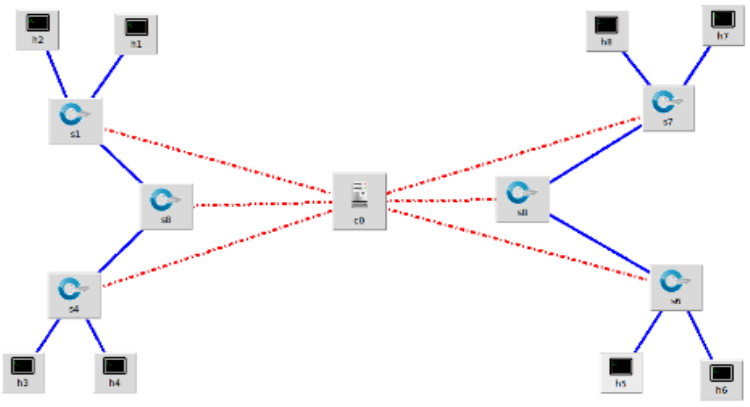
rede local. Assim, em uma das máquinas é instalado o protótipo da ferramenta de avaliação desenvolvido. Na segunda máquina tem-se a instalação dos dois controladores utilizados para os experimentos. E na terceira máquina, tem-se a topologia da rede, à qual é construída através do software de emulação Mininet. Na Figura 15 tem-se o cenário desenvolvido para os experimentos.

Figura 15 - Cenário de Avaliação do Protótipo Desenvolvido
Fonte: do autor.



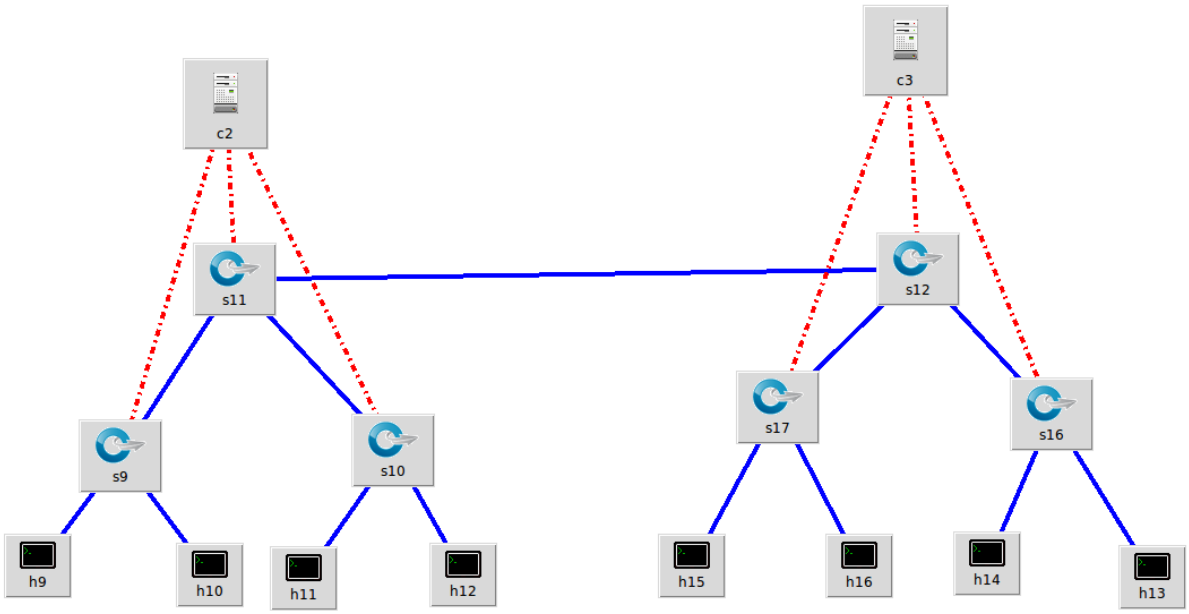
Para realizar os experimentos do protótipo desenvolvido, duas topologias foram definidas a fim de auxiliar às avaliações. Dessa forma, a primeira topologia criada conta com um controlador centralizado, o qual é ligado a seis *switches* virtuais emulados e oito hosts conectados a esses *switches*, conforme ilustra a Figura 16.

Figura 16 - Topologia 1
Fonte: do autor.



A segunda topologia criada conta com dois controladores distribuídos, dessa forma, para que fosse possível efetuar tal análise, foi necessário criar o segundo controlador em uma máquina virtual instalada na máquina hospedeira do primeiro controlador. Desse modo, para essa análise foram conectados a cada controlador três *switches* virtuais e oito hosts conectados aos *switches* (como é ilustrado na Figura 17).

Figura 17 - Topologia 2
Fonte: do autor.



Conforme ilustra as imagens destacadas, ambas as topologias possuem o mesmo número de *switches* e hosts, porém possuem características topológicas diferentes. No primeiro caso tem-se um controlador conectado a cada *switch* virtual. Já no segundo caso, há uma divisão em domínios, no primeiro domínio tem-se apenas 3 *switches* conectados ao controlador 1 e no segundo domínio os outros 3 *switches* são conectados ao controlador 2. Desse modo, o propósito é avaliar cada controlador individualmente e verificar em qual cenário possuem o melhor desempenho.

Assim, inicialmente está planejada a avaliação de dois cenários. O primeiro cenário de avaliação será para medições de desempenho e, o segundo cenário, será voltado para análises de segurança, além disso, para fins de comparação, serão utilizados nos experimentos os três modelos de controladores descritos neste documento. Os cenários definidos para avaliações são apresentados nas subseções seguintes.

7.4 Considerações

O protótipo da ferramenta foi desenvolvido com base nas metodologias pré-definidas pela IETF. Dessa forma, através dos resultados obtidos com essa implementação, foi possível realizar um comparativo com as análises das ferramentas de *benchmarking* analisadas, de modo

a demonstrar como tais ferramentas tornam-se importante para a escolha da correta plataforma de controle e como alterações no ambiente de avaliação podem impactar nestes resultados.

8 RESULTADOS

Este capítulo apresenta os resultados obtidos através das análises feitas sobre as ferramentas apresentadas no Capítulo 6, como também resultados obtidos com o desenvolvimento do protótipo de benchmark de controladores especificado no Capítulo 7. Desse modo, este capítulo está subdivido da seguinte forma: na Seção 8.1 são apresentados os resultados analisados através das ferramentas Cbench e OFCProbe, na Seção 8.2, são apresentadas as análises feitas sobre o protótipo desenvolvido.

8.1 Resultados Obtidos Através das Análises das Ferramentas

Neste capítulo serão apresentados os resultados obtidos através das análises feitas sobre as ferramentas de Benchmark analisadas, resultados esses que serão apresentados nas subseções seguintes:

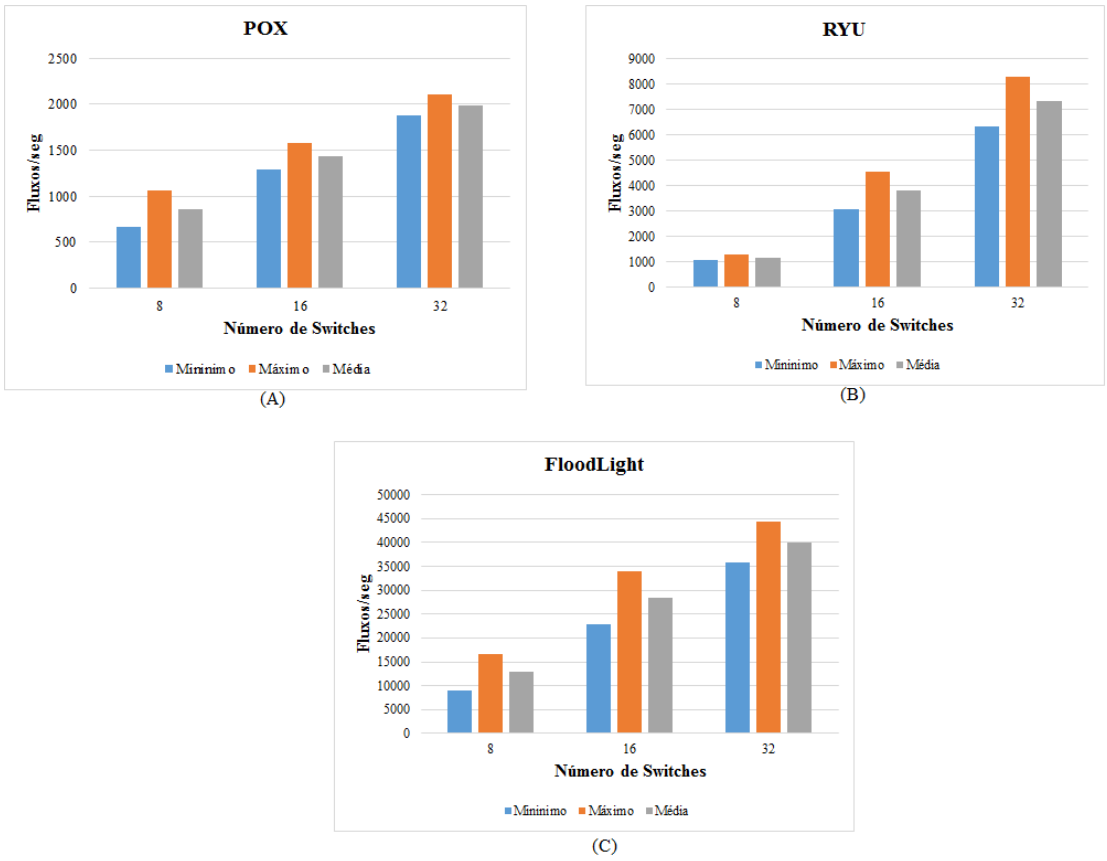
8.1.1 CBench

8.1.1.1 Latência

O modo latência utilizado na ferramenta Cbench não é uma métrica que mede a capacidade de fluxos que o controlador suporta, mas sim o número de fluxos que esse controlador recebe levando em consideração a latência que possui para processar os fluxos de dados recebidos e então solicitar novos fluxos. Dessa forma, neste experimento foram analisados os três controladores descritos nesse documento, o qual foram feitas avaliações com 8,16 e 32 *switches*, analisando assim, basicamente, o comportamento de cada controlador e o tempo de latência para processar os fluxos recebidos pelos *switches* virtuais emulados pela ferramenta Cbench.

Nesta direção, durante os experimentos utilizando essa métrica, todos os controladores avaliados mantiveram-se operando enquanto as avaliações estavam ocorrendo, conforme ilustra a figura 18, na qual exibe um gráfico comparativo com os fluxos de dados recebidos por cada controlador, considerando a latência para processar cada um desses fluxos. Com base nesse gráfico, o controlador Floodlight foi o que obteve um melhor desempenho utilizando essa métrica, conforme pode ser observado no eixo Y dos gráficos, onde mesmo com uma quantidade crescente de *switches* apresentou um melhor suporte a carga de fluxos.

Figura 18 - Fluxos por Segundos Utilizando o Modo Latência
Fonte: do autor.



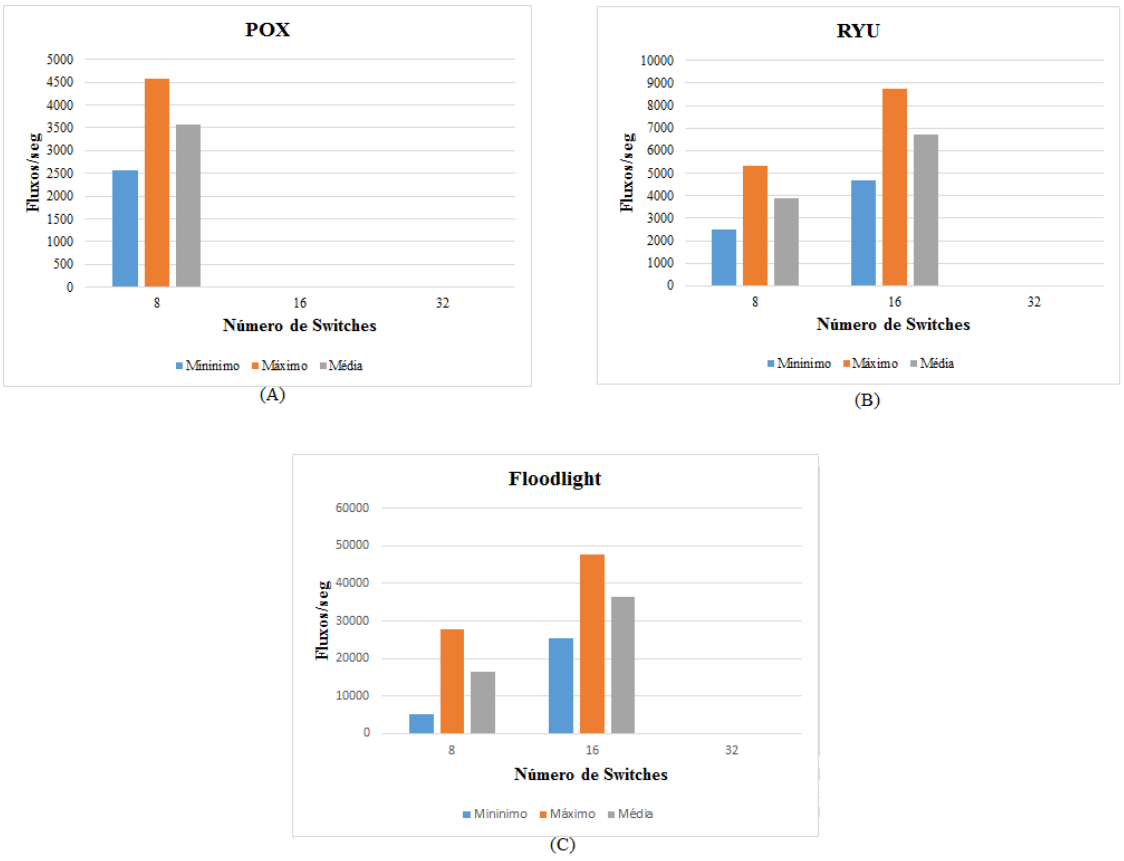
8.1.1.2 Modo Vazão

Neste experimento, o objetivo principal é verificar quantos fluxos cada controlador é capaz de suportar de modo que cada *switch* é possível manter um número alto de solicitações pendentes, ao contrário do que o ocorre no modo latência, na qual apenas é enviada nova solicitação para o controlador quando este processa todas as solicitações já recebidas. Dessa forma, neste modo é possível verificar o limite de solicitações que cada controlador suporta sem ficar inoperante, medindo assim, a taxa máxima de fluxo que pode receber.

Nesta direção, foram realizados experimentos utilizando os cenários já descritos com o intuito de verificar como cada controlador se comporta com um alto nível de fluxos. Dessa forma, no modo vazão, nenhum dos controladores conseguiram suportar o experimento com 32 *switches* virtuais, todos durante a execução desse experimento acabaram ficando inoperantes e o mesmo não pode ser finalizado, conforme é ilustrado na Figura 19, na qual tem-se um gráfico comparativo com os diferentes controladores analisados, informando-se assim, no eixo Y, o número de fluxo de dados por segundo que cada controlador foi capaz de suportar com cada conjunto de *switches*. Desse modo, dentre os controladores analisados, o controlador Floodlight conseguiu ser superior com 8 e com 16 *switches*, e foi o único controlador que suportou duas

iterações utilizando 32 *switches*. Em contrapartida, o controlador POX, conseguiu apenas finalizar o experimento com 8 *switches*, já com 16 *switches* teve um total de 6 iterações, não alcançando o mínimo 10. Por fim, o controlador RYU conseguiu finalizar os experimentos com 8 e com 16 *switches* sendo superior ao controlador POX, porém seus números não são tão expressivos se comparado ao Floodlight.

Figura 19 - Fluxos Por Segundo Utilizando o Modo Vazão
Fonte: do autor.



8.1.2 OFCProbe

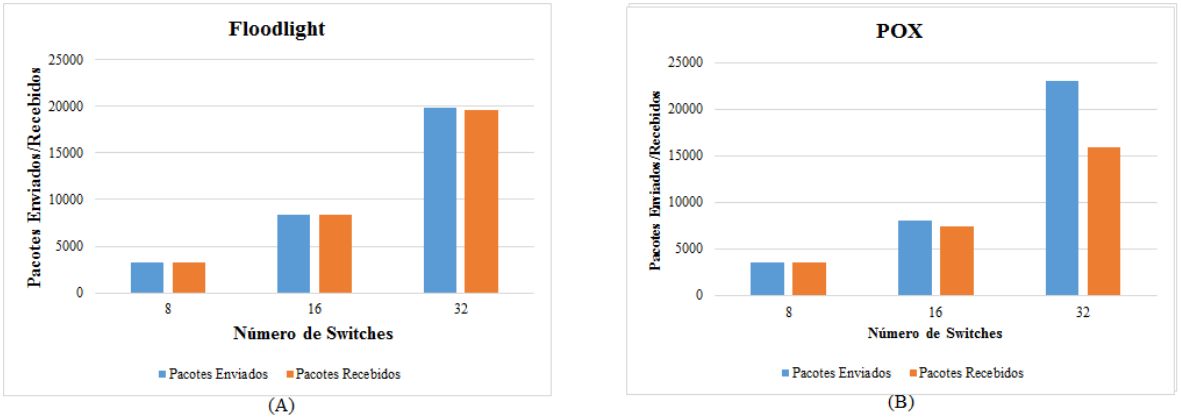
8.1.2.1 Quantidade de Pacotes Enviados e Recebidos

Para esse experimento, foram utilizados apenas os controladores Floodlight e POX, pelo fato de que a ferramenta, utilizando essa métrica, não oferece suporte para o controlador RYU, visto que, durante os experimentos, todos os resultados referentes a quantidade de pacotes enviados por segundos mantiveram-se zerados. Para tanto, os outros dois controladores mostraram compatibilidade com a ferramenta, na qual foi utilizado os parâmetros já descritos no capítulo referente a metodologia das avaliações.

Nesta direção, os experimentos foram feitos em um total de 60 minutos para cada conjunto de *switches*. Dessa forma, foi analisado o total de pacotes enviados por cada dispositivo de encaminhamento comparando-os com o total de pacotes recebidos pelo

controlador, conforme ilustra a figura 20, na qual exibe um gráfico comparativo com a quantidade de pacotes enviados e recebidos no eixo Y, sendo possível analisar como os controladores e os *switches* se comportam e como as características topológicas da rede influenciam nesses resultados.

Figura 20 - Quantidade de Pacotes Enviados e Recebidos Por Segundo
Fonte: do autor.



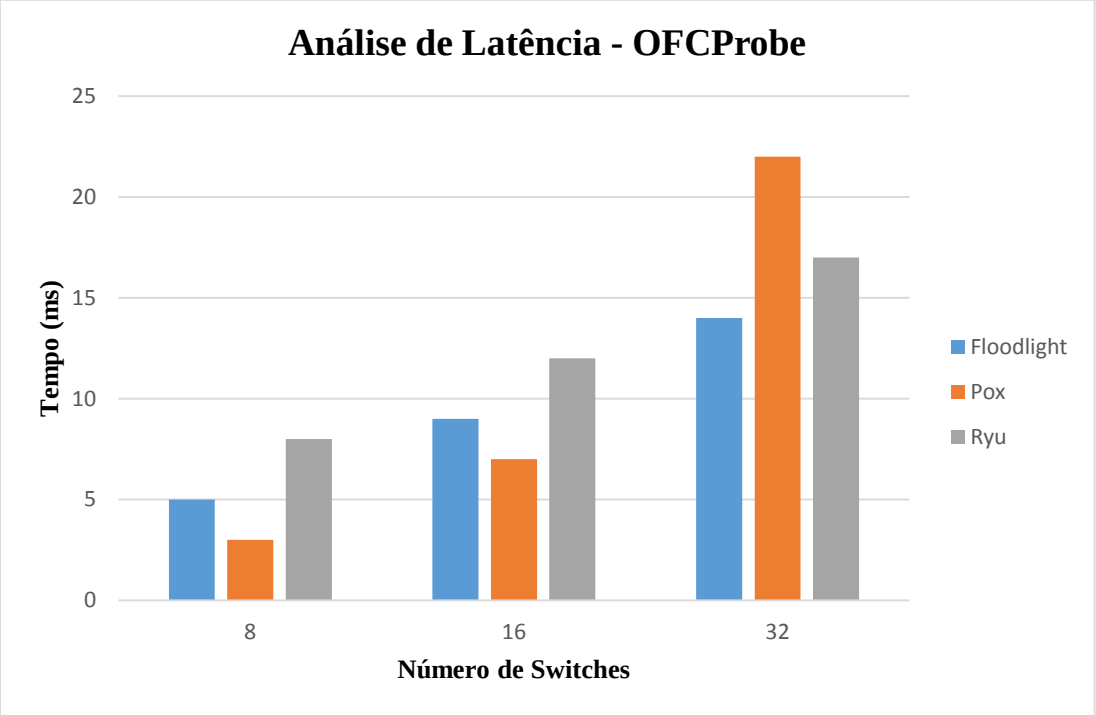
De acordo com as análises feitas através desse experimento, pode-se perceber uma grande diferença entre o controlador Floodlight e o controlador POX, principalmente na transição de 8 para 16 *switches*. O controlador Floodlight manteve uma grande estabilidade entre os pacotes recebidos por ele e os pacotes enviados pelos *switches*, onde pode-se analisar que com 8 *switches* todos os pacotes enviados foram recebidos pelo controlador, ocorrendo o mesmo com o POX, em contrapartida, ao aumentar o número de *switches* na rede para 16, o controlador POX mostrou uma certa instabilidade, onde é possível observar uma certa discrepância entre os pacotes enviados pelos *switches* e os pacotes recebidos pelo controlador, mostrando assim uma quantidade muito superior entre o que é enviado e o que é recebido. Dessa forma, o controlador Floodlight se mostra muito estável nesse quesito, no qual, na pior situação, com 32 *switches*, mostrou que teve uma perda muito pequena dos pacotes enviados e recebidos, perda essa inferior à 3%.

8.1.2.2 Latência

Para esse experimento, foi utilizado o protocolo ICMP (*Internet Control Message Protocol*) para calcular a latência entre os *switches* e o controlador. Dessa forma, para que fosse possível utilizar o controlador RYU, foi necessário apenas uma alteração nas configurações da ferramenta, alterando a porta de comunicação para 6634, porta essa utilizada apenas pelo controlador RYU. Desse modo, nesta avaliação foi possível utilizar os três controladores, como ilustra a figura 21, na qual exibe um gráfico comparativo com os três controladores utilizados para esse experimento, subdivido-os em conjuntos de 8,16 e 32 *switches*. Dessa forma, no eixo

X do gráfico tem-se a subdivisão com os números de *switches* avaliados, e em cada uma dessas, é apresentado cada controlador com seu respectivo tempo de latência na comunicação com os dispositivos de encaminhamento. Esse tempo de latência, representado pelo eixo Y do gráfico, é calculado como a média aritmética feita na comunicação com cada dispositivo de encaminhamento, ou seja, é a divisão da soma dos tempos de latência entre o controlador e cada um dos *switches* pelo total de dispositivos na rede.

Figura 21 - Latência na Comunicação entre o Controlador e os Switches
Fonte: do autor.



Neste experimento é possível perceber uma diferença pouco significante na latência dos *switches* com o controlador, onde é possível observar, nos dois primeiros casos (8 e 16 *switches*), que os três controladores mantiveram uma latência baixa. Porém, quando se aumenta o número de *switches* para 32, o controlador Floodlight mais uma vez mostra ser superior, entretanto, como nesse experimento está sendo utilizado um ambiente controlado, sem interferência da rede externa, a diferença se deve basicamente ao fato de que o controlador Floodight é o único que trabalha com multithread, possibilitando oferecer uma menor latência na comunicação com os *switches*.

8.1.3 Collectl

8.1.3.1 Consumo de Memória RAM

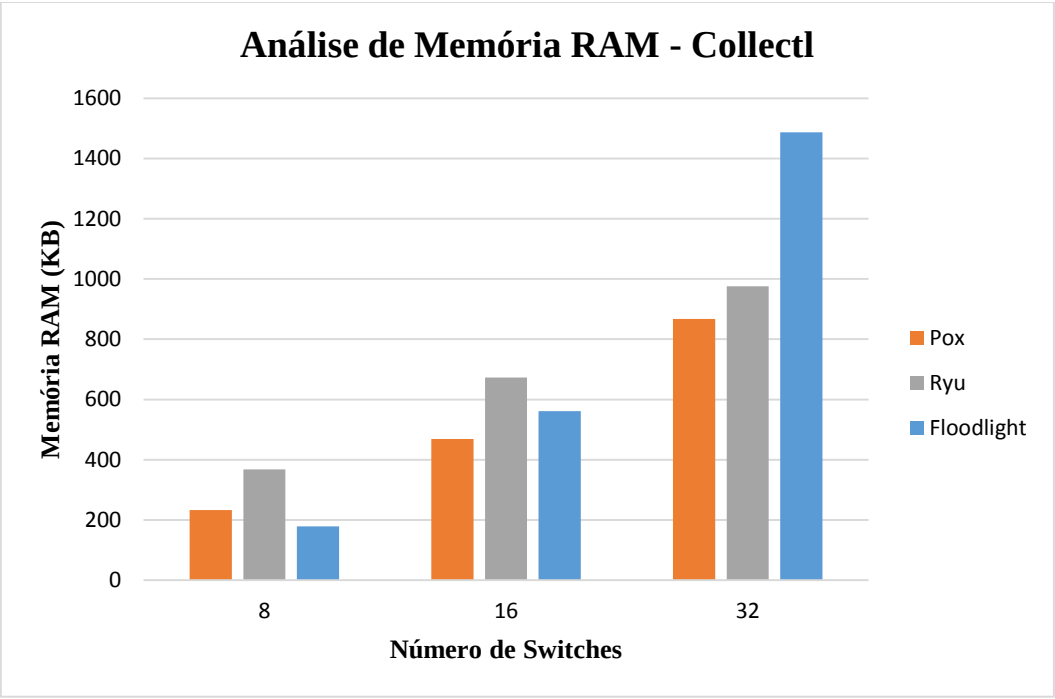
As análises do consumo de memória RAM foram feitas utilizando o software Collectl durante a execução da ferramenta Cbench, a qual é responsável por gerar fluxos de dados sobre os controladores. Dessa forma, cada controlador foi avaliado de forma individual por um

período de 60 minutos para cada um dos experimentos, assim, pode-se analisar como cada um gerencia o consumo de memória RAM com diferentes cenários de rede.

Nesta direção, na Figura 22 tem-se um gráfico comparativo utilizando os três modelos de controladores avaliados neste experimento, o qual exibe o consumo de memória RAM utilizado por cada controlador, sem interferência de outro processo que possa estar sendo executado de forma paralela. Esse consumo de memória é exibido através do eixo Y do gráfico exibido, sendo este apresentado em KB (Kilobytes). Desse modo, o controlador que mostrou um pior gerenciamento de memória RAM foi o controlador Floodlight, ao contrário de todos os outros experimentos no qual mostrava sempre um melhor desempenho, indicando que o fato de ser desenvolvido em Java e ser o único desses controladores a possuir multithread influenciou no resultado obtido. Já o controlador Pox foi o que obteve um menor consumo de memória RAM, suportando assim uma alta carga de fluxo com o consumo de memória baixo.

Apesar disso, como se pode perceber através da escala do gráfico no eixo Y, o consumo de memória relativo é bastante baixo, o que indica que esse não é um problema para essa amostra de controladores, uma vez que o crescimento do consumo de memória é praticamente linear em relação a quantidade de *switches* utilizados.

Figura 22 - Consumo de Memória RAM
Fonte: do autor.



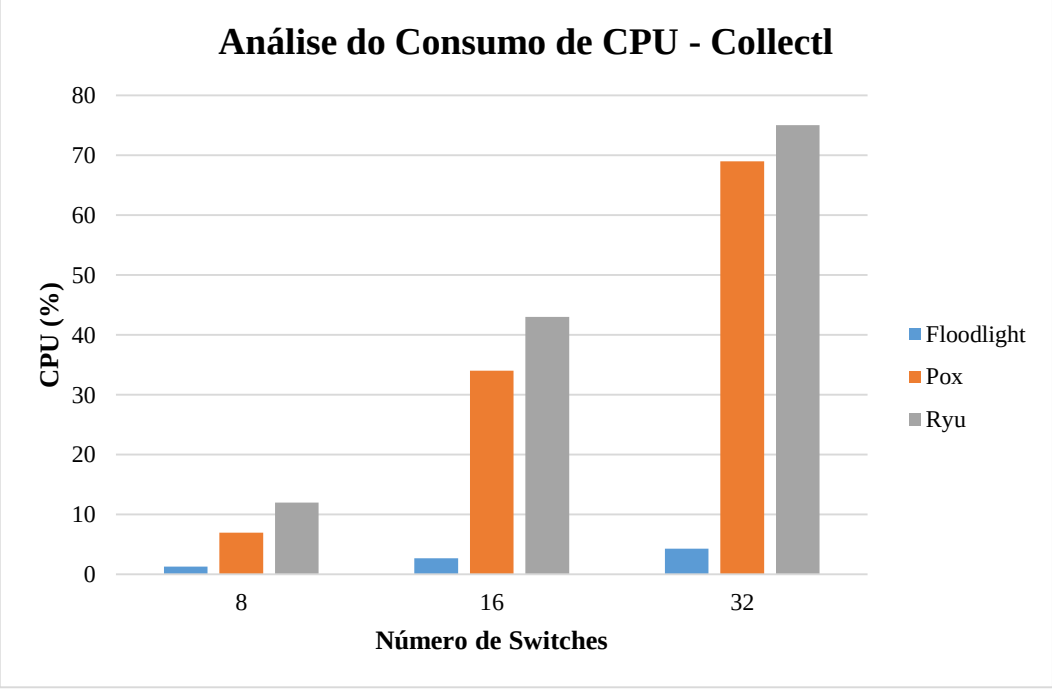
8.1.3.2 Consumo de CPU

Para as análises do consumo de CPU, foram utilizadas as mesmas configurações feitas na análise do consumo de memória. Dessa forma, todo resultado obtido nesse experimento foi

baseado apenas no consumo de CPU do controlador, no qual todos os outros processos e serviços que estavam rodando durante o experimento foram desconsiderados, assim, ao final de cada experimento no total de 60 minutos é exibido o total médio consumido por cada controlador.

Na Figura 23 tem-se um gráfico com as comparações dos três modelos de controladores utilizando os três cenários de experimento definido, ou seja, com 8, 16 e 32 *switches*. Dessa forma, para cada avaliação tem-se o resultado do consumo da CPU em percentual, isto é, consumo referente ao percentual de processamento utilizado da máquina hospedeira, representado no eixo Y do gráfico. Neste experimento executado, o controlador Floodlight obteve um desempenho muito acima da média, o qual durante todo experimento não passou dos 5% de utilização da CPU, ao contrário do que ocorreu na avaliação anterior onde obteve um maior consumo de memória RAM. Em contrapartida, o controlador POX e o controlador Ryu demandaram um consumo de mais de 60% comparado ao Floodlight. Dessa forma, neste experimento fica clara a grande diferença entre o Floodlight e os outros controladores, mostrando assim que o fato de possuir multithread para receber múltiplas instâncias dos *switches* e processá-las paralelamente influencia muito no seu desempenho.

Figura 23 – Consumo de CPU
Fonte: do autor.



8.2 Resultados do Protótipo Desenvolvido

As análises referentes aos experimentos realizados utilizando o protótipo desenvolvido se encontram nesta seção. Dessa forma, os experimentos foram realizados utilizando os dois

cenários descritos no Capítulo 7, Seção 7.3, referente à metodologia de avaliação do protótipo desenvolvido. As duas topologias foram utilizadas a fim de avaliar o comportamento do controlador de acordo com as características planejadas seguindo a IETF, no qual, para cada controlador e para cada cenário foram feitas 10 repetições das avaliações.

Sobre os cenários definidos, foram avaliadas as duas métricas definidas pela IETF: desempenho e segurança. Através da métrica de desempenho, o objetivo principal é avaliar o tempo que leva para o controlador descobrir toda a topologia da rede de acordo com tais características topológicas. Assim, para essa avaliação, o protótipo cliente envia tal requisição para o servidor, o qual recebe o tipo de avaliação que será realizado e solicita para o controlador, solicitação essa feita através de uma interface de comunicação desenvolvida em cada controlador, a qual fica aguardando uma solicitação. Assim, após receber essa solicitação, o controlador executa o método de descoberta da topologia da rede e então descreve essa em um arquivo de texto, de modo que o servidor possa ler e retornar para o cliente com o tempo final da descoberta.

Através da métrica segurança, teve-se como objetivo avaliar a latência do controlador para descobrir a topologia da rede após um ataque DoS, avaliando assim seu comportamento com esse evento, como também o número de requisições que suporta com tal ataque. Assim, para essa avaliação em específico, foi necessário utilizar um módulo separado, módulo esse que efetua os ataques de negação de serviço utilizando o protocolo ICMP para ping. Desse modo, o protótipo cliente envia para o servidor o tipo de experimento que deseja executar e o servidor, ao receber tal solicitação, executa um script escrito em Python que realiza diversas requisições de Ping contra o controlador. Dessa forma, para fins de avaliação, foram definidas 3 estruturas para avaliar a segurança do controlador, as quais serão executadas para verificar seu comportamento. Tais estruturas são descritas na Tabela 4.

Tabela 4 - Estrutura definida para os ataques DDoS.

Protocolo	Número de Hosts	Tamanho do Pacote (bytes)
ICMP	5	65500
ICMP	10	65500
ICMP	15	65500

Dessa forma, teve-se como objetivo principal, avaliar o tempo de descoberta da rede utilizando cada uma das estruturas descritas na Tabela 4. Assim, após o servidor iniciar o script que executa o ataque DoS, o servidor solicita para o controlador a topologia da rede e esse deverá descrevê-la no arquivo de texto para que o servidor possa retorná-la para o cliente e este

avaliar o tempo que levou para tal descoberta, avaliando assim a influência e o comportamento do controlador com tais características na rede.

Nesta direção, os resultados obtidos serão divididos em três subseções de modo a separar os resultados por cada cenário avaliado. Em seguida, após tais análises, será apresentado um quadro comparativo com ambos os resultados.

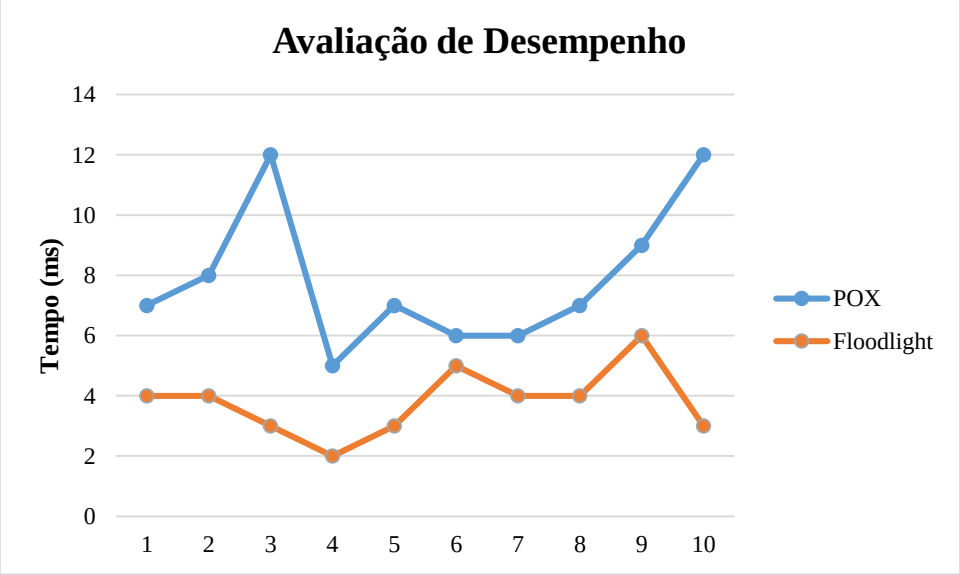
8.2.1 Cenário 1

O primeiro cenário avaliado, consiste em uma rede que possui um controlador centralizado conectado com 6 *switches* virtuais emulados ligados a 8 hosts. Dessa forma, o objetivo é avaliar o comportamento do controlador utilizando essa topologia através das métricas definidas, as quais serão descritas nas subseções seguintes.

8.2.1.1 Desempenho

Os resultados analisados através da métrica de desempenho são apresentados na Figura 24, a qual representa um gráfico com a relação da latência de descoberta da topologia da rede em cada uma das 10 repetições executadas por controlador.

Figura 24 – Tempo de Descoberta da Topologia da Rede (Cenário 1)
Fonte: do autor.

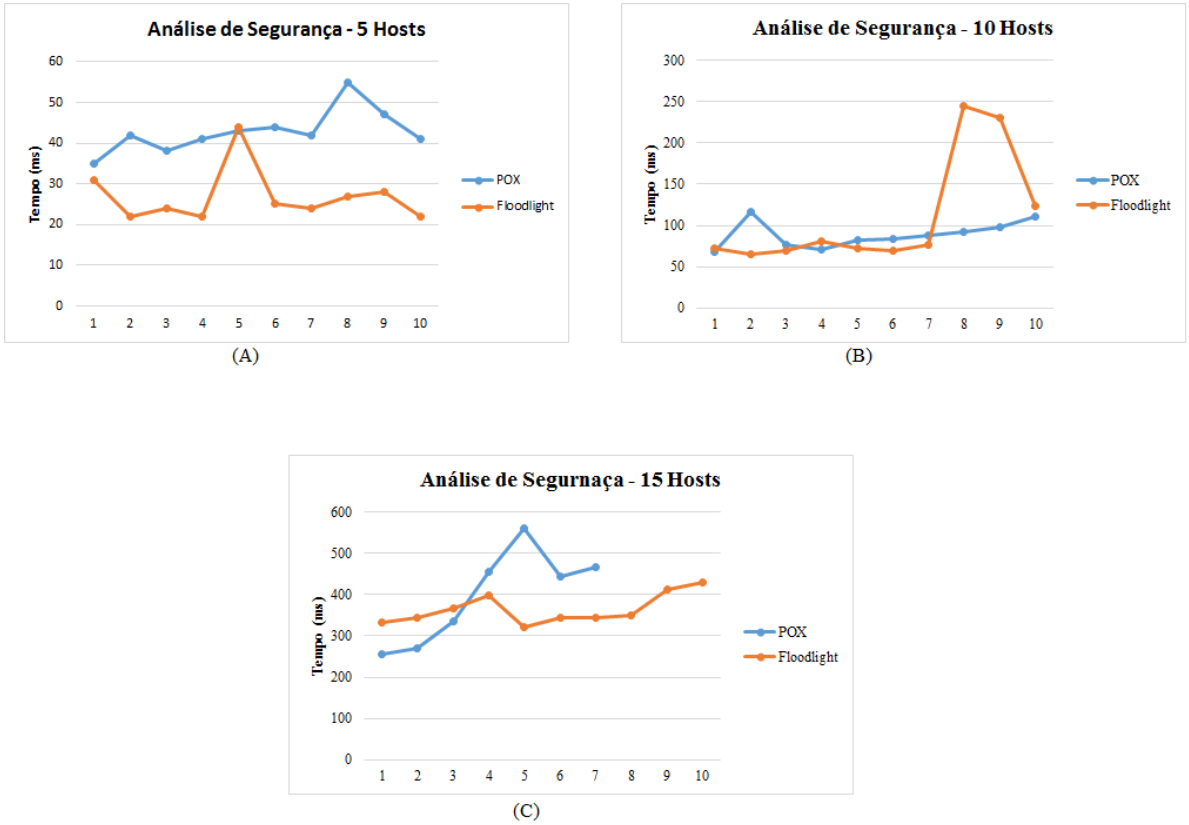


Com base nesse gráfico de latência, pode-se perceber que os controladores analisados mantiveram um tempo baixo para descobrir toda topologia da rede, entretanto, o controlador POX manteve uma certa oscilação no tempo de descoberta, a qual pode ser percebida através das diferenças entre os resultados apresentados por ele. Em contrapartida, o controlador Floodlight manteve um tempo quase que constante, podendo ser observado pelas pequenas diferenças apresentadas em seus resultados, diferenças essas que não ultrapassaram 3 milissegundos.

8.2.1.1 Segurança

Os resultados analisados através dessa métrica de segurança são exibidos na Figura 25, a qual apresenta 3 gráficos de latência, os quais cada um representa o tempo de latência com um determinado número de hosts atacante.

Figura 25 - Análise de Segurança - Cenário 1
Fonte: do autor.



Conforme ilustram os gráficos da análise de segurança na Figura 25, pode-se perceber, analisando o eixo Y dos gráficos, a grande influência no tempo de descoberta da topologia da rede quando um controlador está sobre ataque. No gráfico A, é possível perceber que ambos os controladores mantiveram-se oscilando, ou seja, não conseguiram manter um tempo constante para a descoberta da topologia, porém, ambos obtiveram resultados considerados bons nesse experimento. No gráfico B, o controlador POX se mostrou muito mais estável que o controlador Floodlight, o qual teve um pico na 7ª repetição de 250 milissegundos, porém ambos os controladores se mantiveram disponíveis, suportando tal carga de teste. Enquanto que, no gráfico C, o controlador POX manteve um tempo considerado muito alto para a descoberta da topologia da rede, em vista de que os experimentos são realizados em um ambiente controlado, sem interferência da rede externa, mostrando tal instabilidade a partir da 7ª avaliação, o qual acabou ficando indisponível e não conseguindo mais prosseguir com os testes. Em

contrapartida, o controlador Floodlight manteve-se com uma média de 350 milissegundos para descobrir a topologia da rede, suportando durante todo o experimento os ataques.

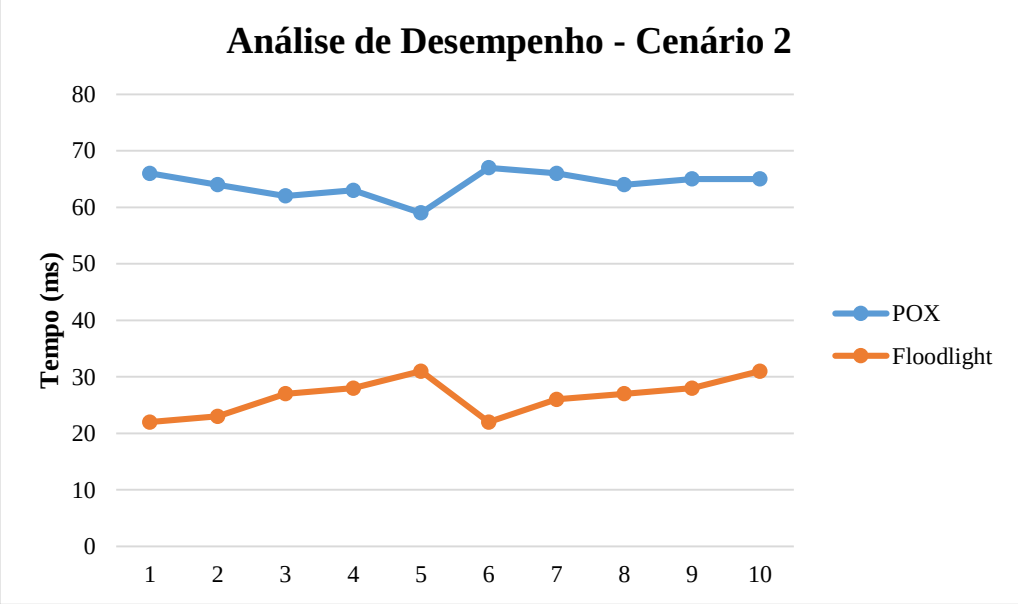
8.2.2 Cenário 2

O segundo cenário avaliado, consiste em uma rede com o mesmo número de dispositivos que a rede avaliada no primeiro cenário. Entretanto, nessa rede tem-se dois controladores distribuídos, de modo que cada controlador fique responsável por gerenciar apenas três *switches* virtuais, conforme a topologia apresentada no capítulo anterior. Dessa forma, o objetivo é avaliar o comportamento dos controladores trabalhando de forma distribuída, de modo que cada controlador, ao ser avaliado, deverá descobrir a topologia de toda a rede, e não apenas do domínio em que está inserido.

8.2.2.1 Desempenho

Os resultados das análises de desempenho são apresentados na Figura 26, a qual apresenta um gráfico ilustrando o tempo de descoberta da topologia da rede por cada controlador.

Figura 26 - Análise de Desempenho - Cenário 2
Fonte: do autor.

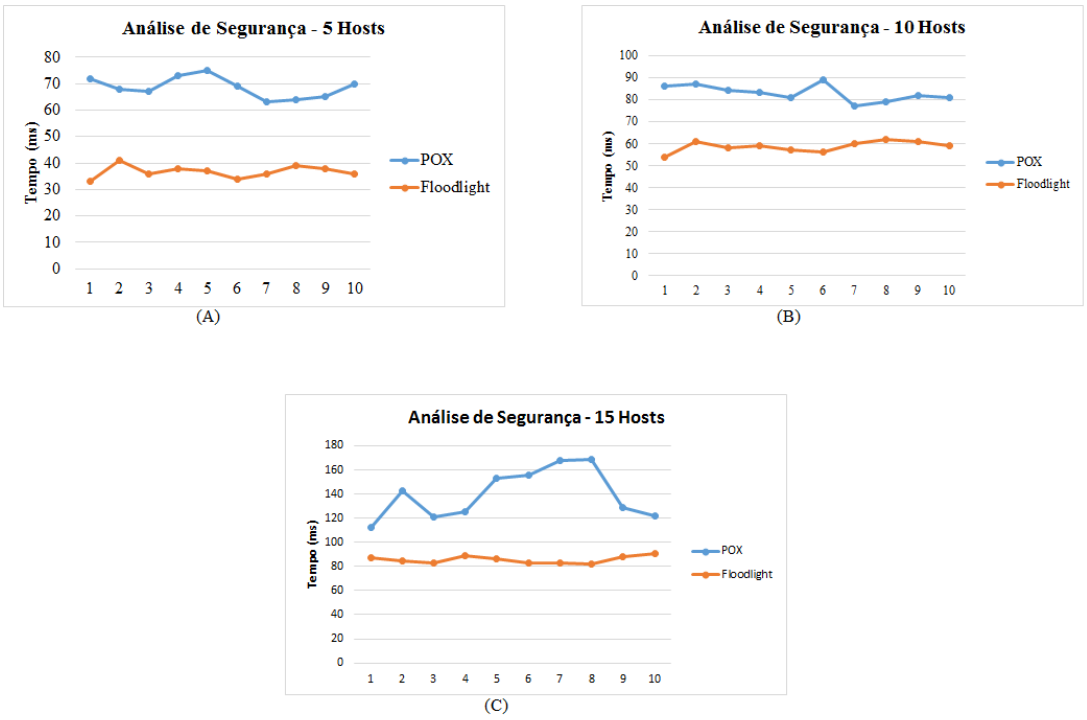


Conforme ilustra o gráfico de desempenho apresentado, todos os controladores avaliados mantiveram uma certa estabilidade na descoberta da rede, em vista de que, cada controlador possuía um número menor de dispositivos para gerenciar. Entretanto, o controlador Floodlight se mostrou superior mais uma vez, no qual teve como tempo máximo para descoberta 51 milissegundos.

8.2.2.2 Segurança

As análises dos resultados de segurança foram feitas da mesma forma que no primeiro cenário, onde teve 3 avaliações por controlador, intercalando assim o número de hosts atacante. Dessa forma, os resultados são apresentados na Figura 27, onde é exibido um gráfico por cada conjunto de hosts atacante.

Figura 27 - Análise de Segurança - Cenário 2
Fonte: do autor.



Conforme ilustra os gráficos apresentados, todos os controladores obtiveram um desempenho muito superior aos apresentados no primeiro cenário, em vista de que, principalmente, todos os controladores mantiveram-se disponível durante toda a execução dos experimentos. O controlador POX, nas duas primeiras avaliações (Figura 27, A e B), manteve um tempo de gerenciamento muito semelhante, onde é possível perceber uma certa estabilidade nos seus resultados, entretanto, ao aumentar para 15 o número de atacantes, este controlador demonstrou uma instabilidade, oscilando de forma considerável seu tempo de descoberta da topologia da rede. Enquanto que, o controlador Floodlight demostrou um desempenho muito superior a todos os outros avaliados no primeiro cenário, conseguindo manter uma constância no tempo de descoberta da topologia da rede em todas as avaliações.

8.2.3 Análise Comparativa dos Resultados Obtidos

Com base nos resultados obtidos através das análises feitas com o protótipo da ferramenta desenvolvida, pode-se perceber como o tempo de descoberta da topologia da rede pode variar de acordo com determinadas características topológicas, como também de acordo com o modo que cada controlador é desenvolvido. Dessa forma, nas tabelas 5 e 6 tem-se quadros comparativos analisando como cada controlador se comporta no momento de descobrir a topologia da rede de acordo com os fatores avaliados em cada um dos cenários (topologias) analisados.

Tabela 5 - Tabela Comparativa com os resultados do cenário 1

Cenário 1			
Controlador	**Avaliação**	**Número de Hosts**	**Tempo médio (ms)**
POX	Desempenho	---	9
	Segurança	5	43
	Segurança	10	89
	Segurança	15	397
Floodlight	Desempenho	----	4
	Segurança	5	27
	Segurança	10	110
	Segurança	15	365

Conforme é apresentado na Tabela 5, pode-se perceber um aumento muito alto no tempo de descoberta da topologia da rede quando o controlador está sobre ataque, sendo possível observar que o tempo para cada um dos experimentos aumenta gradativamente. Apesar do controlador Floodlight manter tempos próximos aos obtidos pelo controlador POX, o controlador Floodlight foi o único que conseguiu finalizar os experimentos com ataques DDoS dos 15 hosts, tornando-se o mais adequado para esse cenário de rede.

Tabela 6 - Tabela Comparativa com os resultados do cenário2

Cenário 2			
Controlador	**Avaliação**	**Número de Hosts**	**Tempo médio (ms)**
POX	Desempenho	---	64
	Segurança	5	68
	Segurança	10	82
	Segurança	15	139
Floodlight	Desempenho	---	26
	Segurança	5	36
	Segurança	10	58
	Segurança	15	85

Com base na Tabela 6 apresentada, é possível perceber que cada um dos controladores analisados mantiveram um tempo de descoberta quase constante em cada experimento, pois o aumento observado no tempo de descoberta da topologia da rede, deve-se basicamente à latência que o controlador tem ao processar as diversas solicitações de ping que recebe, dessa forma, esse aumento há de ser esperado, visto que, todos os controladores mantiveram-se disponível durante todos os experimentos.

9 CONSIDERAÇÕES FINAIS

No contexto atual de SDN, existem diferentes implementações propostas para plataformas de controle. Entretanto, a escolha de uma plataforma de controle deve ser realizada considerando o domínio da rede em que este controlador irá operar. Dessa forma, levando isso em consideração, é clara a importância de se ter um conhecimento prévio sobre as vantagens e desvantagens de cada controlador, pois isso, de certa forma, irá minimizar os conflitos e inadequações após a implantação de uma SDN.

Neste contexto, a utilização de ferramentas que avaliem o desempenho, robustez e confiabilidade dos modelos de controladores torna-se uma etapa de extrema importância para a migração a um modelo SDN. Entretanto, as ferramentas então existentes possuem certas limitações que inviabilizam determinados tipos de experimentos. Dessa forma, utilizar padrões de *benchmarking*, como os fornecidos pela IETF, para avaliar desempenho, escalabilidade, segurança e confiabilidade auxiliam na tomada de decisão para a escolha do controlador correto para cada infraestrutura de rede.

Nesta direção, para esse trabalho de conclusão buscou-se inicialmente efetuar uma avaliação experimental de um conjunto de ferramentas de *benchmark* de controladores, para então, utilizando métricas definidas pela RFC 7426, implementar um protótipo de uma ferramenta de *benchmark* de controladores. Dessa forma, após a implementação do protótipo desenvolvido, a metodologia das avaliações foi aplicada sobre dois controladores, de modo que os resultados obtidos auxiliaram a definir a melhor plataforma de controle para os cenários de rede analisados.

Por fim, são vislumbrados como possibilidades de trabalhos futuros uma análise sobre outras métricas abordadas na RFC 7426 definida pela entidade IETF, buscando assim aumentar a quantidade de experimentos analisados como também o número de plataformas de controladores avaliadas. Dessa forma, aumenta-se o escopo de experimentos a serem realizados, possibilitando assim comparar outras características de cada controlador. Além disso, outros pontos poderão ser tratados como possibilidades de trabalhos futuros, os quais são destacados a seguir:

1) Executar os experimentos analisados com outros parâmetros a serem definidos, como também aumentar o número de *switches* por controlador bem como o número de hosts por *switches.*
2) Aplicar outras técnicas de análises estatísticas em busca de uma maior precisão sobre os resultados analisados.

3) Analisar o comportamento dos controladores em outros cenários de avaliação, alterando principalmente a topologia da rede bem como o número de controladores.
4) Efetuar os experimentos em um ambiente real, para que sejam analisados os diversos recursos avaliados nesse trabalho, buscando assim ratificar os resultados obtidos.

REFERÊNCIAS

HAKIRI, A.; GOKHALE, A.; BERTHOU, P.; Schmidt, D.C.; GAYRAUD, T. “Software-Defined Networking: Challenges and Research Opportunities for Future Internet”, Computer Networks, Volume 75, Part A, p. 453-471, 2014.

CARIA, M.; JUKAN, A.; HOFFMANN, M. SDN Partitioning: A Centralized Control Plane for Distributed Routing Protocols, IEEE Transactions on Network and Service Management, p. 381-393, 2016.

CASADO, M.; KOPONEN, T.; RAMANATHAN, R.; SHENKER, S. Virtualizing the Network Forwarding Plane, Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow, SIGCOMM ACM Comput. Commun, p. 1-6, 2010.

FARIAS, L.; DINIZ, M.; LUCENA, S. Uma Abordagem para Redução da Tabela de Encaminhamento sob a Ótica da Interface de Saída dos Pacotes, XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação, CSBC 2014. Rev, 2014.

FERNANDEZ, M. Evaluating OpenFlow Controller Paradigms, The Twelfth International Conference on Networks, ICN 2013.

GENGE, B.; HALLER, P. A Hierarchical Control Plane for Software-Defined Networks-based Industrial Control Systems, IFIP Networking Conference (IFIP Networking) and Workshops, p. 73-81, 2016.

HONG, C.; KANDULA, S.; MAHAJAN, R.; ZHANG, M.; GIL, V.; NANDURI, M.; WATTENHOFER, R, Achieving High Utilization with Software-Driven WAN, ACM SIGCOMM, p. 15-26, 2013.

JAIN, S.; KUMAR, A.; MANDAL, S.; ONG, J.; POUTIEVSKI, L.; SINGH, A.; VENKATA, S.; WANDERER, J.; ZHOU, J.; ZHU, M.; ZOLLA, J.; HÖLZLE, U.; STUART, S.; VAHDAT, A. B4: Experience with a Globally-Deployed Software Defined WAN, ACM SIGCOMM, p. 3-14, 2013.

JARSCHEL, M.; LEHRIEDER, F.; MAGYARI, Z.; PRIES, R. A Flexible OpenFlow-Controller Benchmark. Proceedings of the European Workshop on Software Defined Networking, p. 48-53, 2012.

JARSCHEL, M.; METTER, C.; ZINNER, T. OFCProbe: A platform-independent tool for OpenFlow controller analysis, Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on, p. 182-187, 2014.

KAUR, S.; SINGH, J.; GHUMMAN, N. International Conference on Communication,Computing & Systems, At SBS State Technical Campus , Ferozepur, Punjab , India, 2014.

KREUTZ, D.; RAMOS, F. M. V.; VERÍSSIMO, P.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-Defined Networking: A Comprehensive Survey, Communications Surveys Tutorials, IEEE, p. 14-76, 2014.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks, Proceeding Hotnets-IX Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p. 1-6, 2010.

LEVIN, D.; CANINI, M.; SCHMID, S.; SCHAFFERT, F.; FELDMANN, A. Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks, USENIX ATC, p. 333-345, 2014.

LEVIN, D.; WUNDSAM, A.; HELLER, B.; HANDIGOL, N.; FELDMANN, A. Logically Centralized? State Distribution Trade-offs in Software Defined Networks, SIGCOMM Comput. Commun. Rev, p. 1-6, 2012.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, p. 69-74 2008.

NADEAU, T. D.; GRAY, K.SDN: Software Defined Networks. 1. ed. United States of America: Kristen Borg, 2013.

NETWORK, Big Switch. Floodlight. Disponível em: <https://projectfloodlight.org/floodlight>. Acesso em: 20 set. 2016.

NUNES, A. B.; MENDONÇA, M.; NGUYEN, N. X.; OBRACZKA, K.; TURLETTI, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, Communications Surveys Tutorials, IEEE, p. 1617-1634, 2014.

ONF. OpenFlow Switch Specification. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf/>. Acesso em: 15 out. 2016.

ONF. SDN Architecture. Disponível em: <https://www.opennetworking.org>. Acesso em: 5 ago. 2016.

OpenFlow, POX Wiki. POX. Disponível em: <https://openflow.stanford.edu/display/ONL/POX+Wiki>. Acesso em: 10 out. 2016.

Ryu Documentation Release 4.15, October 28, 2016. Disponível em: <https://media.readthedocs.org/pdf/ryu/latest/ryu.pdf>. Acesso em: out. 2016.

SCHMID, S.; SUOMELA, J. Exploiting Locality in Distributed SDN Control, SIGCOMM Comput. Commun. Rev, p. 121-126, 2013.

SEZER, S.; SCOTT-HAYWARD, S.; FRASER, B.; LAKE, D.; FINNEGAN, J.; VILJOEN, N.; MILLER, M.; RAO, NAVNEET.Are We Ready for SDN? Implementation Challenges for Software-Defined Networks, Communications Maganize, IEEE, p. 36-43, 2013.

SHAMILOV, A.; ZUIKOV, D.; ZIMARINA, D.; PASHKOV, V.; SMELIANSKY, R. Advanced Study of SDN/OpenFlow controllers, CEE-SECR, p. 1-6, 2013.

STRINGER, J.; PEMBERTON, D.; FU, Q.; LORIER, C.; NELSON, R.; BAILEY, J.; CORREA, C.; ROTHENBERG, C. Cardigan: SDN Distributed Routing Fabric Going Live at

an Internet Exchange, Computers and Communication (ISCC), p. 1-7, 2014 IEEE Symposium on.

TOOTOONCHIAN, A.; GANJALI, Y. HyperFlow: A Distributed Control Plane for OpenFlow, Proceedings of the 2010 internet network management conference on Research on enterprise networking, USENIX Association, p. 3-3, 2010.

TOOTOONCHIAN, A.; GORBUNOV, S.; GANJALI, Y.; CASADO, M.; SHERWOOD. On Controller Performance in Software-Defined Networks, in Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association, p. 10-10, 2012.

VENGAINATHAN, B.; BASIL, A.; TASSINARI, M.; MANRAL, V.; BANKS, S. RFC 7426: Benchmarking Methodology for SDN Controller Performance, draft-ietf-bmwg-sdn-controller-benchmark-meth-02, 2016.

VENGAINATHAN, B.; BASIL, A.; TASSINARI, M.; MANRAL, V.; BANKS, S. RFC 7426: Benchmarking Methodology for SDN Controller Performance, draft-ietf-bmwg-sdn-controller-benchmark-term-02, 2016.

VISSICCHIO, S.; VANBEVER, L.; BONAVENTURE, O. Opportunities and Research Challenges of Hybrid Software Defined Networks, SIGCOMM Comput. Commun. Rev, p. 70-75, 2014.

Collectl Documentation Release 4.2.0, June 12, 2017. Disponível em:
< http://collectl.sourceforge.net/index.html>

Santa Cruz do Sul, 22 de junho de 2017.

_____

Cassiano de Mello Padilha

_____

Prof. Me. Lucas Fernando Müller
Orientador