FACULDADE VALE DO SALGADO
CURSO REDES DE COMPUTADORES

RANGEL HENRIQUE FÉLIX

ANÁLISE DA APLICAÇÃO RYU EM UM AMBIENTE SDN: UM EXPERIMENTO UTILIZANDO O PROTOCOLO OPENFLOW E O MININET

ICÓ-CE

2019

RANGEL HENRIQUE FÉLIX

ANÁLISE DA APLICAÇÃO RYU EM UM AMBIENTE SDN: UM EXPERIMENTO UTILIZANDO O PROTOCOLO OPENFLOW E O MININET

Projeto apresentado a Faculdade Vale do Salgado/FVS, como requisito para a obtenção de aprovação do Curso de Redes de Computadores com ênfase em Segurança da Informação.

Orientador: Prof. João Carlos da Cruz de Lima.

ICÓ-CE

2019

RANGEL HENRIQUE FÉLIX

ANÁLISE DA APLICAÇÃO RYU EM UM AMBIENTE SDN: UM EXPERIMENTO UTILIZANDO O PROTOCOLO OPENFLOW E O MININET

Projeto apresentado a Faculdade Vale do Salgado/FVS, como requisito para a obtenção da aprovação do Curso de Redes de Computadores com ênfase em Segurança da Informação.

Data da aprovação: ____/____/____

BANCA EXAMINADORA:

_____

Prof. Adriano Lima Cândido
Faculdade Vale do Salgado - FVS
(Orientador)

_____

Prof.
João Carlos da Cruz de Lima - FVS
(1º Membro)

_____

Prof.
Francisco Silvanê Nogueira Saldanha - FVS
(2° Membro)

RESUMO

A Internet levou à criação de uma sociedade digital em rede, onde quase tudo está conectado e é acessível de qualquer lugar. No entanto, apesar de sua adoção generalizada, as redes IP tradicionais são complexas e muito difíceis de gerenciar. É difícil configurar a rede de acordo com as políticas definidas e reconfigurá-la para responder as falhas. O paradigma Redes Definidas por Software (*Software* Defined Network, SDN), é um dos temas relacionados a redes de computadores muito discutidos em eventos nacionais e internacionais nos últimos anos. Esse interesse se deve às diversas possibilidades de aplicação do paradigma, que apresenta-se como uma forma potencial de implantação do que se convencionou denominar Internet do Futuro, além disso, a SDN é um modelo emergente que promete mudar esse estado de negócios, separando a rede lógica do controle dos roteadores e *switches* subjacentes, promovendo centralização do controle de rede e introdução da capacidade de programar a rede. O presente artigo traz uma análise sobre a aplicação do controlador Ryu, com o propósito de demonstrar as possibilidades da combinação de teste em um ambiente simulado de SDN. Para realização dos testes, foi criado dois cenários com o Mininet e injetado um tráfego na rede. Através de experimentos realizados foi possível perceber que o Ryu controlou os ambiente como era esperado e que a complexidade da infraestrutura não causou grandes diferença de atraso nas transferências de dados.

Palavras Chave: SDN. OpenFlow. Mininet. Controladores de rede. Ryu

ABSTRACT

The Internet has led to the creation of a networked digital society where almost everything is connected and accessible from anywhere. However, despite their widespread adoption, traditional IP networks are complex and very difficult to manage. It is difficult to configure the network according to the defined policies and reconfigure it to respond to failures. The Software Defined Network (SDN) paradigm is one of the topics related to computer networks much discussed at national and international events in recent years. This interest is due to the various possibilities of applying the paradigm, which presents itself as a potential way of implementing what is known as the Internet of the Future. Moreover, SDN is an emerging model that promises to change this state of business, separating the logical control network of the underlying routers and *switches*, promoting centralized network control and introducing the ability to program the network. This paper presents an analysis on the application of the Ryu controller, with the purpose of demonstrating the possibilities of the test combination in a simulated SDN environment. To perform the tests, two scenarios were created with Mininet and injected a network traffic. Through experiments, it was possible to see that Ryu controlled the environments as expected and that the complexity of the infrastructure did not cause large differences in data transfer delays.

Key words : SDN. OpenFlow. Mininet. Network controllers. Ryu

1 INTRODUÇÃO

O paradigma Redes Definidas por *Software* (SDN), é um dos temas relacionados a redes de computadores muito discutidos em eventos nacionais e internacionais nos últimos anos. Esse interesse se deve às diversas possibilidades da aplicação do paradigma, que apresenta-se como uma forma potencial de implantação do que se convencionou denominar o futuro da internet (GUEDES et al., 2012).

A maioria das atividades que a sociedade realiza hoje passa, de alguma forma, por uma ou mais redes de computadores. Com o advento da internet das coisas, onde terá um grande aumento de dispositivos conectados às redes, a limitação e a complexidade das operações tendem a aumentar. Dessa forma, as redes precisarão ser mais flexíveis, escaláveis, programáveis, seguras e ter uma melhor disponibilidade. A forma como as redes são operadas é um problema, pois são complexas e difíceis de gerenciar. Essas redes possuem vários tipos de equipamentos, tais como, roteadores, *firewalls*, balanceadores de carga de servidor, *switches* e os mesmos executam em sua maioria *softwares* de controle fechados e proprietários e passam anos para criarem uma padronização e interoperabilidade (FEAMSTER et al., 2014). Além disso, os administradores configuram dispositivos de rede individualmente usando interface de configuração que variam entre fornecedores e até em diferentes produtos do mesmo fornecedor.

A SDN é hoje uma solução disruptiva que está mudando a forma de projetar redes, pois tem controle separado da parte lógica de dados, para que seja fácil gerenciar a arquitetura de rede em um cenário virtual (SATASIYA e RUPAL, 2016). Controle dinâmico e decisão de lidar com tráfego permite interrupções sem incorrer sobrecarga de manutenção. Plano de controle diferencial e plano lógico permitem facilitar implantação de novas aplicações, além disso tem gestão simplificada do protocolo. SDN é uma arquitetura futurista que é poderosa fácil de gerenciar, menos dispendioso e flexível (KAUR et al., 2015). Sendo assim, já é uma realidade testada por grandes projetos, podemos citar como exemplo o B4 da Google, onde implementaram SDN para interligar data centers de todo o planeta e com isso alavancaram a velocidade bruta dos servidores e atualizaram os servidores independentemente do *hardware* do *switch* (JAIN et al., 2013).

Diante do contexto apresentado, este trabalho visa analisar e verificar o desempenho do controlador de rede Ryu em ambientes SDN por meio de testes de vazão, bem como expor características de funcionamento do mesmo, além disso, a pesquisa faz uma abordagem sobre estruturas de redes definida por *software*. Dada a sua importância disruptiva, as SDNs e as

aplicações de rede desenvolvidas baseadas no protocolo OpenFlow[1] são hoje alvo de grandes investimentos em pesquisas (DE OLIVEIRA et al., 2017) . Estas pesquisas avançam para o estado da arte com soluções inovadoras que trazem diversos benefícios para o setor acadêmico e privado. No entanto, para que estas vantagens sejam de fato reais, a avaliação ser consistentes. A motivação para a escolha do controlador Ryu foi Devido ao uso da linguagem de programação Python[2] e suporte às versões 1.0, 1.2, 1.3 e 1.4 do protocolo OpenFlow, bem como a classificação *Analytic Hierarchy Process* (*AHP*) proposta por (KHONDOKER et al., 2014), optou-se pelo controlador Ryu para servir de controlador da rede nos testes.

As próximas seções estão organizadas da seguinte forma: a Seção 2 trata do referencial teórico; a Seção 3 descreve os tipos de ambientes de experimentação e a dificuldade de escolha por meio da metodologia; a Seção 4 apresenta os resultados e discussões sobre a avaliação de desempenho realizada, descrevendo os objetivos e a configuração dos ambientes de teste detalhadamente; e por fim a Seção 5 apresenta as conclusões e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Redes Definidas Por Software

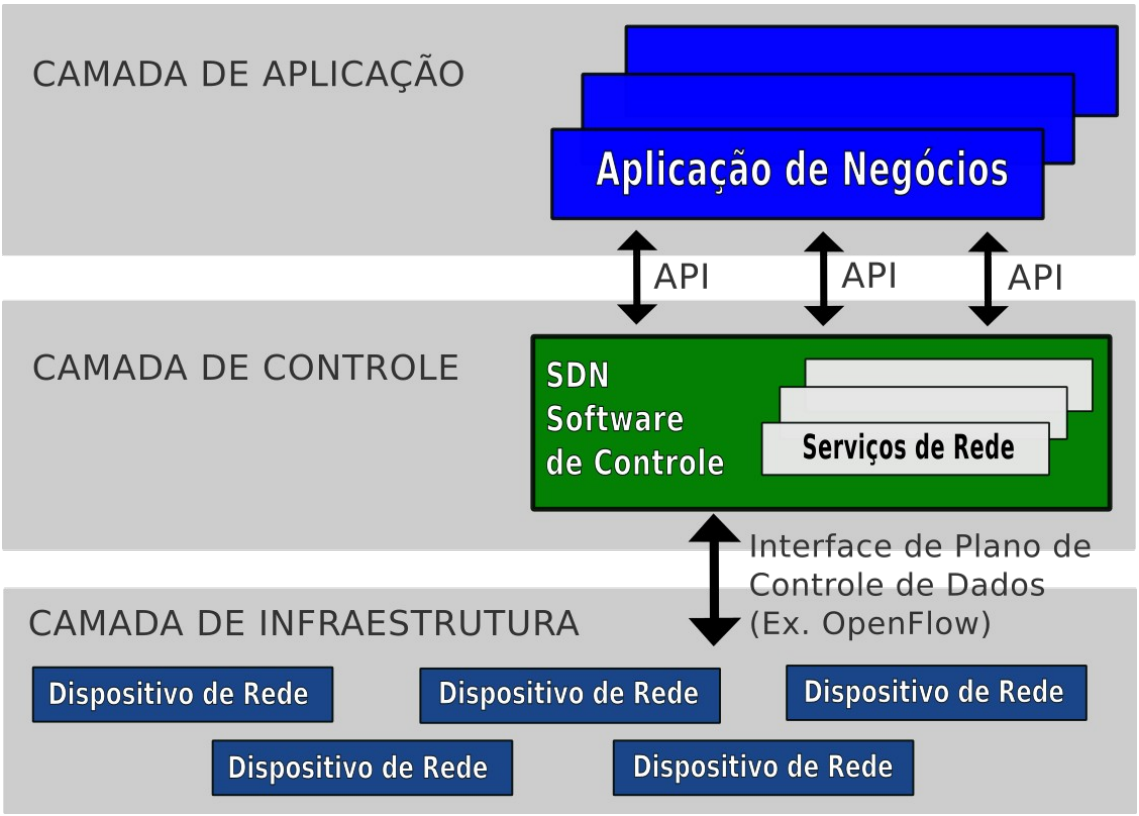
A Rede definida por *Software* (SDN), está mudando a forma como as redes são projetadas e gerenciadas. O conceito por trás da SDN teve sua evolução desde 1996, impulsionado pelo desejo de fornecer aos usuários gestão controlada do encaminhamento nos nós da rede. Implementações realizadas por grupos de pesquisadores e da indústria, como por exemplo, Ipsilon (1996), The Tempest (1998), Internet Engineering Força-Tarefa Internacional - IETF (2000), Caminho Elemento de Computação (2004), e mais recente Ethane (2007) e OpenFlow (2008), trouxeram a SDN para mais perto da realidade (SEZER et al., 2013). As redes de computadores estão em constante evolução e precisam ser flexíveis, escaláveis programáveis e mais seguras e ter uma melhor disponibilidade (LIMA et al., 2018).

Em conformidade com Boucadair e Jacquenet (2015), a SDN é um conjunto de métodos utilizados para simplificar o design, a entrega e a operação de serviços de rede de forma determinística, dinâmica e de forma escalável. Na arquitetura SDN, os planos de controle e dados são desacoplados, a inteligência de rede e o estado são logicamente centralizados, e a infraestrutura subjacente é abstraída das aplicações.

1 Site oficial do projeto: https://www.opennetworking.org
2 Site oficial do projeto: https://www.python.org

A SDN concentra-se em quatro características principais: separação do plano de controle do plano de dados; um controlador centralizado e visão de rede; abrir interfaces entre os dispositivos do plano de controle (controladores) e aqueles no plano de dados e capacidade de programação da rede por aplicações (ONF, 2012). Ainda em consonância com a ONF (2012), a SDN é uma arquitetura de rede emergente onde o controle é dissociado do encaminhamento e é diretamente programável. Esta migração de controle, antigamente rigidamente ligada em dispositivos de rede, permite que a infraestrutura seja abstraída para aplicações e serviços e pode tratar a rede como uma entidade lógica ou virtual, como podemos ver na Figura 1.

Figura 1. Arquitetura funcional SDN ilustrando os elementos de infraestrutura, controle e aplicação da qual a rede é composta.



Fonte: Adaptada de ONF (2012).

A Figura 1 representa uma visão lógica da arquitetura SDN, onde a inteligência de rede é (logicamente) centralizada em controladores SDN baseados em *software*, que mantêm uma visão global da rede. Como resultado, a rede aparece para os aplicativos e mecanismos de políticas como um único *switch* lógico. Com SDN, empresas e operadoras ganham controle independente de fornecedor sobre toda a rede a partir de um único ponto lógico, o que simpli-

fica bastante o projeto e operação da rede. Além do mais, também simplifica muito os dispositivos de rede, já que não precisam mais entender e processar milhares de padrões de protocolo, mas apenas aceitam instruções dos controladores SDN (ONF, 2012).

As definições e descrições do funcionamento e das aplicações de Redes Definidas por Software, são encontradas na literatura por diversos autores, sendo assim, de acordo com Feamster, Rexford e Zegura (2014), SDN parte do princípio da separação de um plano de controle logicamente centralizado a partir do plano de dados subjacente. Do mesmo modo para Yeganeh, Tootoonchian e Ganjali (2013), afirmam que mover a função de controle dos elementos para fora do plano de dados é o denominador comum entre propostas de SDN para a comunidade de pesquisa. Este desacoplamento permite que ambos os planos evoluem de forma independente, e traz sobre inúmeras vantagens, como alta flexibilidade, independência de fornecedor, capacidade de programação e a possibilidade de realizar uma rede centralizada Visão (YEGANEH; TOOTOONCHIAN; GANJALI, 2013).

Apesar de todas essas vantagens, serem preocupações sobre o desempenho e escalabilidade desde a sua criação. A percepção comum de que o controle na SDN é centralizado leva a preocupações com a escalabilidade da SDN. Afinal, independentemente do capacidade do controlador, um controlador central não escalar à medida que a rede cresce (aumentar número de comutadores, fluxos, largura de banda, etc.) e não conseguirá lidar com todas as solicitações recebidas enquanto fornecendo as mesmas garantias de serviço (YEGANEH; TOOTOONCHIAN; GANJALI, 2013).

2.2 Protocolo OpenFlow

Segundo Rothenberg et al., (2010), o OpenFlow foi proposto pela Universidade de Stanford no ano de 2007 para atender à necessidade de validação de novas propostas de arquiteturas e protocolos de rede sobre equipamentos comerciais. Em conformidade com Fernandes e Rothenberg (2014), OpenFlow é uma tecnologia nova e de rápida evolução que habilita a implementação de Redes Definidas por *Software*. A arquitetura do OpenFlow define um modelo de rede no qual os elementos encaminhadores, chamados de comutadores OpenFlow, são simples e programáveis (MATTOS et al., 2011). Neste mesmo sentido para McKeown et al., (2008), o OpenFlow surge como uma forma de padronização da configuração por meio de um controle unificado de comutadores, roteadores e pontos de acesso sem fio.

Em suma, o principal objetivo do OpenFlow é ter um sistema de controle que proporciona o desenvolvimento com softwares, capazes de definir e alterar a comutação da rede através de uma interface (Farias et al., 2011), eliminando restrições em privilégios de autorização do acesso, estabelecendo função de fluxos entre *hosts* baseadas nos atributos do fluxo, tornando-se compatível com o maior número de equipamentos possível.

De acordo com (GUEDES et al., 2012), a versão mais recente do OpenFlow é a 1.1, que ainda possuem limitação em termos do uso do padrão em circuitos ópticos e uma definição de fluxos que englobe protocolos que não fazem parte do modelo TCP/IP. No entanto, a versão 2.0 está sendo formulada e um dos objetivos é eliminar algumas dessas limitações. Alguns *switches* comerciais já suportam o padrão OpenFlow, como é o caso do HP Procurve 5400zl, do NEC IP880 e do Pronto 3240 e 3290. Espera-se que à medida que a especificação OpenFlow evolua, mais fabricantes de equipamentos de rede incorporem o padrão às suas soluções comerciais.

2.3 Controladores de Rede

Controladores de Rede são *softwares* desenvolvidos para o gerenciamento da camada de controle da arquitetura de redes definidas por *software*, que facilitam o controle e a abstração para aplicar configurações específicas de encaminhamento e controle de pacotes. São eles que mantém as informações da topologia e monitoram o estado total da rede, tudo em um só lugar, propagando a configuração para os demais nós. Promove ainda uma interface para criar, modificar e controlar a sua tabela de fluxos dos comutadores. Consoante com Rodrigues e Guimarães (2014), um controlador Openflow é responsável por todo o conjunto de *switches* Openflow; é executado ao lado do servidor. Por conseguinte, o Openflow desempenha meios de controlar os dispositivos sem a necessidade dos fabricantes demonstrarem o código fonte dos produtos, tornando assim o controlador a peça principal do processo. Dentre os diversos controladores existentes podemos citar como exemplo: Nox, Pox, Ryu, Floodlight, Mul, Maestro, Trema, dentre outros, como podemos ver na Tabela 1 abaixo:

Com o tempo foram criados muitos outros controladores OpenFlow, onde os mesmos utilizam diferentes plataformas e linguagens de programação. Na Tabela 1, podemos ver os principais controladores utilizados em ambientes SDN, bem como as principais características de cada um. É importante lembrar que a escolha de um controlador influencia diretamente no

tempo de processamento do plano de controle, porém o estudo realizado neste trabalho não foca na comparação de controladores OpenFlow, más visa analisar o comportamento do controlador Ryu em ambientes SDN.

Tabela 1. Tipos de controladores SDN

Controlador	Implemetação	Livre	Desenvolvedor	Visão Geral
POX	Python	SIM	Nicira	Controlador SDN geral.
NOX	Python/C++	SIM	Nicira	O primeiro controlador Openflow.
MUL	C	SIM	Kulcloud	Possui uma infra-estrutura multi-thread.
Maestro	Java	SIM	Rice University	Fornece interfaces de controle modulares.
Trema	Ruby/C	SIM	NEC	Desenvolvimento Openflow.
Beacon	Java	SIM	Stanford	Utiliza uma plataforma modular.
Jaxon	Java	NÃO	Independent Developers	Baseado no NOX.
Helios	C	SIM	NEC	Fornece um shell programático.
Floodlinght	Java	NÃO	BigSwitch	Switches físicos e virtuais Openflow.
SNAC	C++	SIM	Nicira	Disponibiliza um interface Web.
Ryu	Python	SIM	NTT, OSRG group	Controle logicamente centralizado.
NodeFlow	JavaScript	SIM	Independent Developers	Um controlador Openflow JavaScript.
Ovs-controller	C	SIM	Independent Developers	Gerencia qualquer número de switches.
FlowVisor	C	SIM	Stanford/Nicira	Permite múltiplos controladores na rede física.
RouteFlow	C++	SIM	CPqD	Realiza roteamento IP virtual.

Fonte: Adaptada de (Jain et al. 2013)

2.4 Controlador Ryu

O controlador Ryu[3] é um *software* de código aberto baseado em componentes definido por uma estrutura de rede. Ele é implementado inteiramente em Python e suportado pelos laboratórios da NTT[4]. Como outras estruturas de controlador SDN, o Ryu também fornece componentes de *software* com APIs bem definidas que são expostos para permitir que os desenvolvedores criem novos aplicativos de gerenciamento e controle de rede (RAO, 2015).

A escolha do Ryu como controlador para realização dos teste desse trabalho é baseado no modelo proposto por Khondoker et al., (2014), onde em seu trabalho selecionou o Ryu por ser um controlador que tem uma excelente documentação e uma implementação bem elaborada e foi elaborado para uso geral. Outros pontos fortes do controlador podem ser vistos em Kreutz (2014), no qual diz que o Ryu é um dos poucos controladores que é compatível com as versões 1 à 3 do protocolo OpenFlow. Em harmonia com os autores acima, Rao (2015), defende que um dos pontos fortes do Ryu é que ele suporta vários protocolos para o gerenciamento de dispositivos, como o OpenFlow, o Network Configuration Protocol (NETCONF), o OpenFlow Management e o Configuration Protocol (OF-Config), entre outros.

3 Site oficial do projeto: https://github.com/osrg/ryu

4 Site oficial: https://www.ntt.co.jp/index_e.html

Segundo Rao (2015), Assim como qualquer controlador SDN, o Ryu também pode criar e enviar uma mensagem OpenFlow, ouvir eventos assíncronos e analisar e manipular pacotes de entrada.

3 METODOLOGIA

Nesta seção, são apresentados experimentos com o propósito de analisar o desempenho do controlador Ryu. É realizada a execução da liberação de fluxo e uma análise da vazão, além disso, serão feitos testes de desempenho com sobrecarga de tráfego.

Pesquisar e avaliar o desempenho de redes de computadores, envolve a análise de métricas que caracterizam seu comportamento, desde a sua topologia física (fim a fim) quanto todas as especificações operacionais e projetos lógicos (DINIZ e JÚNIOR, 2014). Para isto devem ser feitas escolhas corretas e análises de métricas nos diferentes níveis funcionais de uma rede. Essas métricas podem ser obtidas através de técnicas de medição ativa que consistem na injeção de pacotes de controle na rede, em que são apenas observados os pacotes que trafegam pela rede. As características da rede que são geralmente medidas e analisadas pelas ferramentas para a análise de desempenho de rede são: Largura de banda e taxa de uso dos canais de comunicação envolvidos (*throughput* ou vazão); Perda de pacotes; Tempo de resposta (latência ou atraso da rede); *Jitter*: é obtido como o desvio padrão do atraso de pacotes enviados em sequência (DINIZ e JÚNIOR, 2014).

Para a obtenção de resultados experimentais em ambiente simulado, um cenário SDN foi montado, utilizando uma máquina virtual com Ubuntu 14:04 configurada com 2GB de RAM, 2vcpu, 50GB dinâmico, já com o Mininet 2.2.2 e o controlador Ryu 4.33, instalados na plataforma VirtualBox[5] em um *host* com 4GB RAM, 4cpu, 250GB SSD e sistema operacional *LINUX* UBUNTU 19.04, x86_64, Intel(R) Core(TM) i3-5005u 2.00GHZ. A escolha de controladores distintos influenciam no tempo de processamento do plano de controle, no entanto essa pesquisa não foca na comparação de controladores OpenFlow.

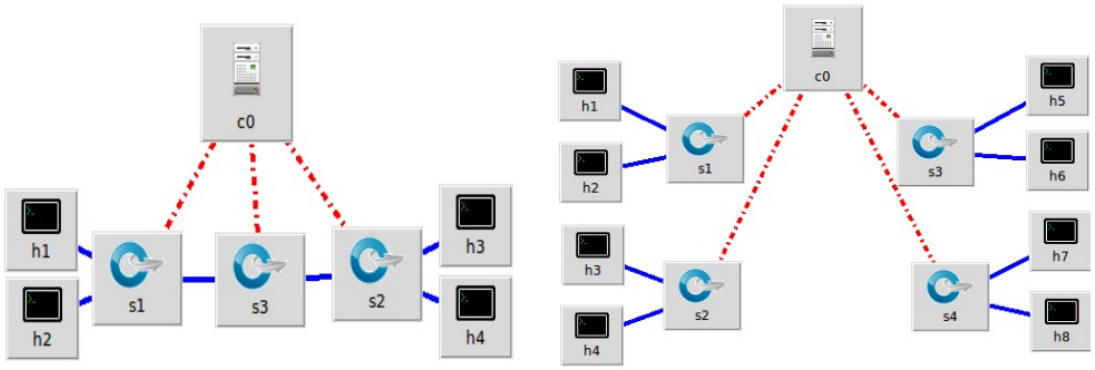
Como emulador de rede escolhemos o Mininet por ser amplamente difundido e usado para pesquisas em SDNs. Mininet permite ao pesquisador recriar redes de dados com a im-

5 Site Oficial Oracle VirtualBox em https://www.virtualbox.org.

plementação real do padrão topologias (por exemplo, árvore, árvore gorda, Bcube, linear, dentre outras) e protocolos por exemplo STP, ECMP (Ortiz et al., 2016).

Para realizar os testes, foi utilizado o gerador de tráfego Iperf (V2.0.5)[6]. O desempenho do controlador foi medido usando o Mininet ambiente de simulação de rede definido por software. Com o Mininet foi criada duas topologias, uma em árvore e uma linear. A topologia em árvore é composta com a seguinte configuração: 4 *switchs* virtualmente conectados com um único controlador e 8 *hosts* virtuais, 2 *hosts* com cada *switch*, já os testes do cenário linear foram feitos com a seguinte configuração: 3 *switches* conectados virtualmente com um controlador e 4 *hosts* virtuais. A topologia em árvore é composta com a seguinte configuração: 4 *switchs* virtualmente conectados com um único controlador e 8 *hosts* virtuais, 2 *hosts* com cada *switch*, já os testes do cenário linear foram feitos com a seguinte configuração: 3 *switches* conectados virtualmente com um controlador e 4 *hosts* virtuais, como podem ser vistos na Figura 2, ambas topologias foram criadas utilizando o (miniedit)[7] do Mininet.

Figura 2. Topologias: tipo Árvore e Linear usadas para simulação no Mininet.



a) cenário tipo linear *b) cenário tipo árvore*

Fonte: Autor.

A medição da banda foi feita através do Iperf transmitindo pacotes UDP com valor de largura de banda igual a 1 GB, durante aproximadamente 30 segundos, utilizando a porta 5201 e monitorando os resultados a cada segundo. Utilizou-se então os comandos abaixo nos *hosts* servidor e cliente, respectivamente: iperf -s -u -p 5201 -i 1 -t 30 e iperf -c “ip servidor” -p 5201 -i 1 -t 30 -b 1g. Os detalhes do funcionamento dos comandos e uma abordagem mais detalhada do uso do Iperf pode ser vista em (DINIZ e JUNIOR, 2014).

6 Site Oficial IPerf em https://iperf.fr.
7 Usando Miniedit: http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/

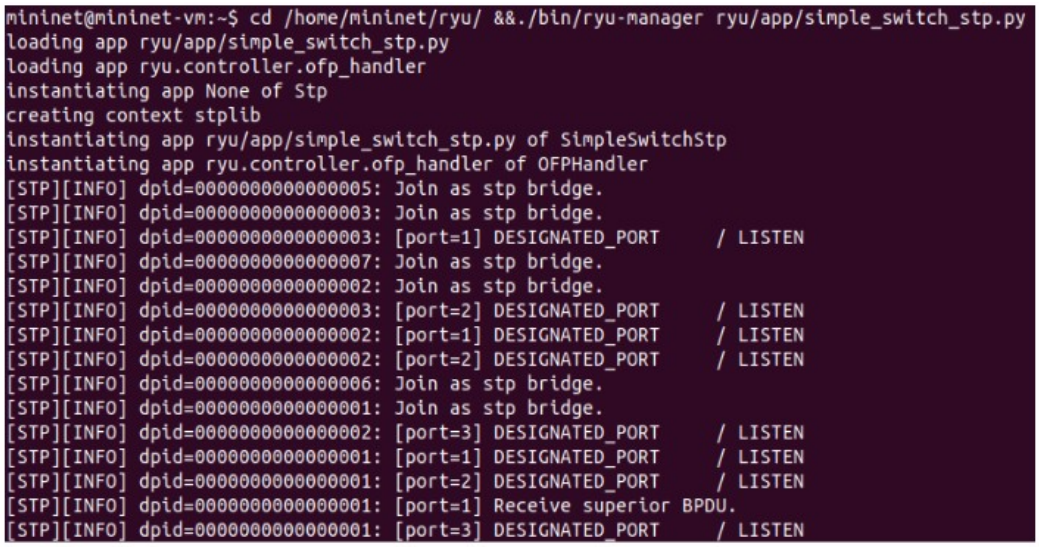
4 ANÁLISE DOS RESULTADOS

Com os cenários mencionados na secção anterior, foi possível fazer os testes. Utilizando o *pingall* comando do Mininet foi gerado um tráfego que executa testes de ping entre cada *host* na rede emulada. Isso gera tráfego para o controlador toda vez que um *switch* recebe um pacote que possui um endereço MAC de destino que ainda não está em sua tabela de fluxo (Linkletter, 2015). Em seguida foi realizada uma análise de desempenho da rede. Os cálculos de vazão foram feitos com o programa Iperf, que é uma ferramenta utilizada para obter resultados referentes às taxas de transferência de dados entre os *hosts*. Por não possuir interface gráfica, suas execuções acontecem por linhas de comandos (Iperf, 2019). A análise foi feita observando as taxas de tráfego nos dois cenários.

No momento que a topologia árvore/linear criada no Mininet está tendo conexão entre *switch* e *hosts*, cada *host* se conecta com seu *switch* específico e os *switches* são conectados entre si. Todos os *switches* estão habilitados para OpenFlow, por sua vez, são conectados a um controle remoto. Agora, o controlador Ryu fica ativo no mesmo endereço IP remoto (127.0.0.1), o mesmo comando de linha única é executado para criar a topologia linear que conecta o controlador Ryu em vez de um controlador nativo. Na figura 3 podemos ver o momento que o controlador é ativado após a topologia ser criada.

Fazendo uma análise da figura acima, percebemos que o controlador Ryu está recebendo pacotes da camada de dados, e o Protocolo Spanning Tree Protocol (STP), pode ser executado e mostra o estado da porta do *switch*. Quando o STP está ativado, a porta do *switch* começa em BLOCK e depois muda para o estado LISTEN e LEARN. O estado BLOCK é uma condição que a porta não participa do encaminhamento de quadros e descarta o quadro recebido do segmento de rede. Durante o estado LISTEN, o *switch* processa quadros e aguarda possíveis novas informações que possam fazer com que retorne ao estado de bloqueio. No estado LEARN, a porta começa a processar o quadro e inicia a atualização da tabela de endereços MAC. O estado FORWARD é o estado normal pelo qual o quadro passou pela porta.

Figura 3. Status do controlador Ryu conectando um comutador das topologias.



Fonte: Autor.

Os dpid são identificadores que o algoritmo do STP cria para o envio dos dados. Na figura 3, o estado do STP está em LISTEN, pois é o momento em que o tráfego foi gerado e o protocolo descartou o segmentos de outros links desnecessário, eliminando loops (IRAWATI e NURUZZAMANIRRIDHA, 2015).

4.1 Análises do Controlador Ryu na topologia do tipo Linear

Nesta topologia todos os *switches* e *hosts* foram habilitados para OpenFlow e estão conectados uns com os outros de forma linear. Uma arquitetura de detalhes e o trabalho da topologia linear é discutido em (Rakesh et al, 2014). Após ativar um controlador Ryu no mesmo endereço IP remoto (127.0.0.1), um comando Mininet é executado para gerar rede na topologia em árvore que conecta o controlador Ryu no endereço IP atribuído, como pode ser visto na Figura 3. O ip servidor foi substituído pelo o ip da máquina correspondente ao servidor. Esse mesmo teste foi realizado em H1 e H3 como servidor e aos demais *hosts* como cliente. Os resultados podem ser observados na Tabela 2.

Tabela 2. Testes de vazão utilizando os servidores (H1 e H3)

H1 SERVER				
Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perca
H2	2,48	792	0,001	2.3%
H3	2,54	790	0	8,70%
H4	2,74	785	0,003	0,87%
Total/ Média	2,59	789	0,0013	4,79%
H3 SERVER				
Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perca
H4	2,71	793	0,001	2,30%
H2	2,76	790	0,002	0,48%
H1	2,75	788	0,14	0,65%
Total/ Média	8,22	790,33	0,0477	1,14%

Fonte: Autor.

Para verificar os resultados de forma separada em cada *host* da rede foi utilizado o *Xterms*. O terminal *Xterm* se conecta a um *host* na rede virtual, sendo basicamente usado para executar um comando interativo e assistir a saída de depuração (LIMA et al., 2018). No momento em que o tráfego é gerado o controlado Ryu começa a controlá-lo, como podemos ver na Figura 4.

Figura 4. Status do controlador Ryu no momento do tráfego com iperf (Linear).

```
[STP][INFO] dpid=0000000000000002: Join as stp bridge.
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000002: [port=2] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000002: [port=1] Receive superior BPDU.
[STP][INFO] dpid=0000000000000002: [port=1] DESIGNATED_PORT     / BLOCK
[STP][INFO] dpid=0000000000000002: [port=2] DESIGNATED_PORT     / BLOCK
[STP][INFO] dpid=0000000000000002: [port=3] DESIGNATED_PORT     / BLOCK
```

Fonte: Autor.

O STP escutando o tráfego de rede, após o término da solicitação ele bloqueia o encaminhamento de pacotes e fica aguardando novas instâncias. Neste caso como o controlador está conectado em topologia linear o STP elege as portas que servirão para o encaminhamento dos pacotes, em seguida as portas são bloqueadas para evitar um loop futuro.

Para verificar melhor a performance do controlador, os dados da tabela 2 foram condensados com tempos e taxas médias de transferência de dados, esses valores podem ser vistos na Tabela 3.

Tabela 3. Taxas Médias de vazão utilizando os servidores (H1 e H3)

Servidores	Transferências (GB/s)	Transferência em (Mb/s) / Bandwidth	Jitter	Perca
H1	2,59	789	0,0013	4,79%
H3	8,22	790,33	0,0477	1,14%

Fonte: Autor.

4.2 Análises do controlador Ryu na topologia do tipo Árvore

Em uma topologia de rede em árvore, todos os *switches* e *hosts* habilitados para OpenFlow estão conectados uns com os outros de forma hierárquica. Uma arquitetura mais detalhada e o trabalho da topologia em árvore é discutido em (Rakesh et al., 2014). Após ativar um controlador Ryu no mesmo endereço IP remoto (127.0.0.1) que foi discutido acima, um comando Mininet de linha única é executado para gerar rede na topologia em árvore que conecta no controlador Ryu no endereço IP atribuído. O ip servidor foi substituído pelo o ip da máquina correspondente ao servidor. Esse mesmo teste foi realizado em H1, H3, H5 e H7, como servidor e aos demais *hosts* como cliente. Os resultados podem ser observados na Tabela 4.

Para uma melhor interpretação dos dados de forma separada em cada *host* da rede foi utilizado o *Xterms*. No momento em que o tráfego é gerado o controlado Ryu começa a controlá-lo, como podemos ver na Figura 5.

Para uma rede Ethernet funcionar corretamente, apenas um caminho ativo pode existir entre duas estações. Como a topologia em árvore por ser um pouco mais complexa podemos ver o protocolo STP atuando nas transferências geradas com o iperf. Abrindo conexões com a porta do comutador e escutando o tráfego (LISTEN), e após a transferência a porta é

bloqueada (BLOCK), evitando assim os fluxos desnecessários. Como neste exemplo foram feitos vários testes de encaminhamento o algoritmo do STP aprende a porta de encaminhamento dos quadros, facilitando o envio futuro de novos pacotes.

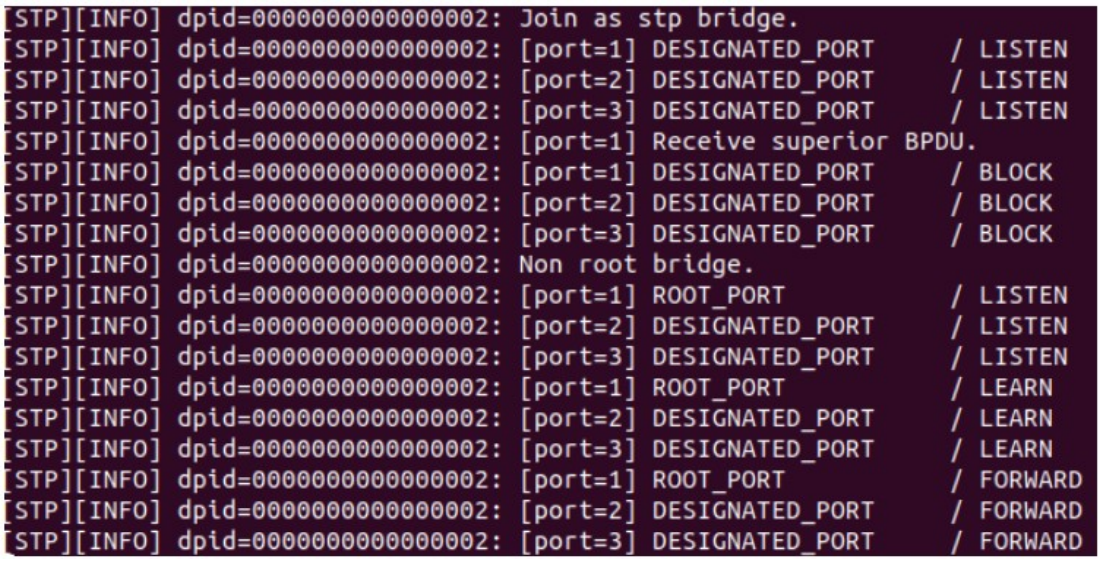
Tabela 4. Testes de vazão utilizando os servidores (H1, H3, H5 e H7)

	Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perda
H1 SERVER	H2	2,76	793	0,001	1%
	H3	2,73	783	0,007	0,83%
	H4	2,76	793	0,001	1,00%
	Total/ Média	2,75	790	0,0030	0,94%
	Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perca
H3 SERVER	H4	2,73	785	0,001	1,40%
	H2	2,75	791	0,000	1,30%
	H1	2,74	787	0,076	1,30%
	Total/ Média	8,22	788	0,0257	1,33%
	Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perca
H5 SERVER	H2	2,72	780	0	0,85%
	H3	2,66	764	0,001	1,00%
	H4	2,74	785	0,003	0,87%
	Total/ Média	2,71	776	0,0013	0,91%
	Host	Transferência em (GB/s)	Transferência em (Mb/s)	Jitter (ms)	Perca
H7 SERVER	H4	2,77	796	0,001	1,22%
	H2	2,76	790	0,002	0,48%
	H1	2,75	788	0,14	0,65%
	Total/ Média	8,28	791	0,0477	0,78%

Fonte: Autor.

Para verificar melhor a performance do controlador, os dados da Tabela 2 foram condensados com tempos e taxas médias de tranferência de dados, esses valores podem ser visualizados na Tabela 5.

Figura 5. Status do controlador Ryu no momento do tráfego com iperf (Árvore).



Fonte: Autor.

Tabela 5. Taxas Médias de vazão utilizando os servidores (H1, H3, H5 e H7)

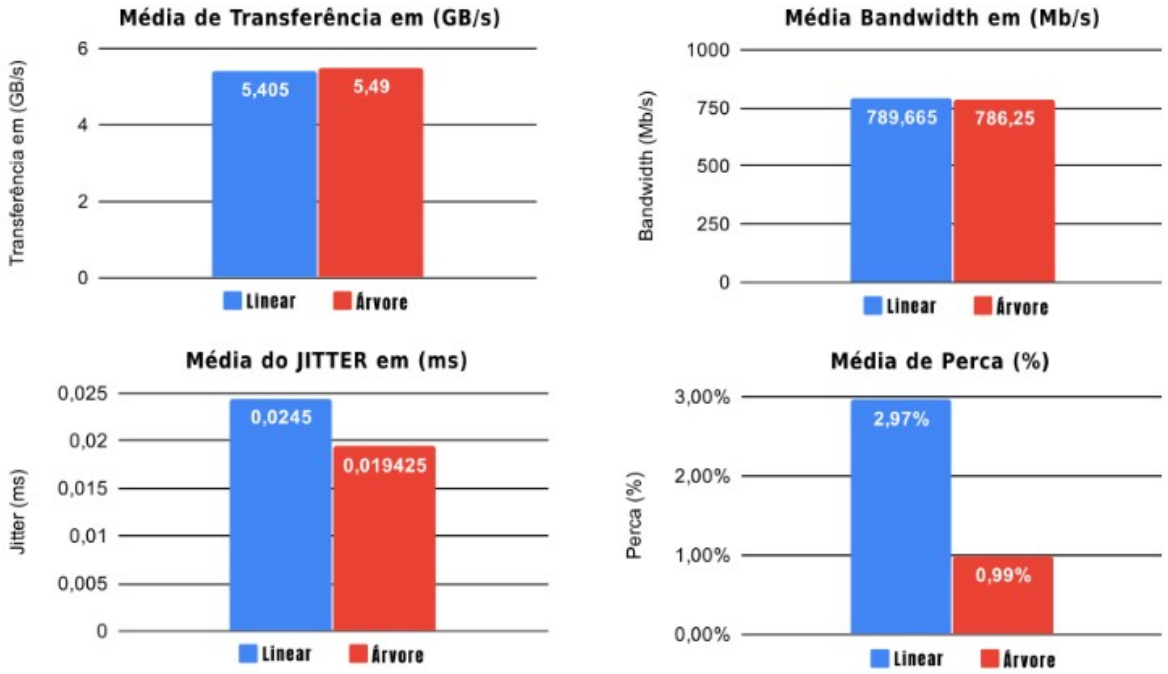
Servidores	Transferências (GB/s)	Transferência em (Mb/s) / Bandwidth	Jitter	Perca
H1	2,75	790	0,003	0,94%
H3	8,22	788	0,0257	1,33%
H5	2,71	776	0,0013	0,91%
H7	8,28	791	0,0477	0,78%

Fonte: Autor.

4.2 Análises das duas Topologias

Uma comparação de desempenho entre a topologia de rede linear e árvore definidas anteriormente são alcançadas pela execução do teste de vazão comparando a taxa de tranferência, variação de atraso (Jitter) e uma porcentagem de perda. Na Figura 6, temos um gráfico de comparação e com ele podemos verificar se as infraestruturas poderão ou não influenciar no desempenho da rede.

Figura 6. Análise de desempenho nas Topologias Linear x Árvore.



Fonte: Autor.

Podemos perceber na Figura 6 que a taxa média de transferência em ambas topologias mantiveram-se bem próximas com uma pequena variação de 0.085 GB/s a mais da topologia em árvore. Por sua vez, o Bandwidth também se manteve com uma pequena diferença de 3.415 Mb/s de transmissão de quadros. Já a taxa média do Jitter teve uma diferença de 0.005075 ms, esse valor se tratando de topologias simples pode não ser tão relevante, mas em estruturas mais complexa isso pode afetar o desempenho da rede. Fazendo uma análise da perca de quadros, a topologia linear teve uma perca bem maior o que afeta também um arquitetura em funcionamento.

5. CONCLUSÃO

As redes SDN baseadas em OpenFlow é uma tecnologia promissora para o futuro da internet. Neste artigo, o comportamento de controlador Ryu foi avaliado através de experimentações baseadas em tráfego de vazão em cenários do tipo árvore e linear, ambos com o protocolo OpenFlow configurado nos *switches* de rede. A escolha do ambiente é uma das premissas para se alcançar um resultado mais fiel. Uma vez que o desempenho do

ambiente afeta diretamente o resultado da validação de novas propostas, que avaliem o tempo de processamento do ambiente, vazão, atraso, entre outros. A utilização do ambiente Mininet pode ser uma excelente escolha para o início de aprendizagem sobre o funcionamento do OpenFlow e do conceito de SDN, tendo a vantagem de ser uma máquina virtual obtida diretamente em sites especializados e facilmente colocada em qualquer computador com um *software* de virtualização.

Com base na comparação de desempenho entre as topologias criadas, o controlador, mostrou ser capaz de controlar a rede, bem como separar o plano de controle do plano de dados, foi possível observar que o controlador Ryu está recebendo os dados de transferência da infraestrutura, dessa forma podemos comprovar os princípios da SDN onde o controlador consegue recuperar informações das camadas de dados e posteriormente essas informações podem ser manipuladas em uma camada de aplicação, além disso oferecendo desempenho eficiente em todos os testes de vazão realizados nos dois cenários. A complexidade da topologia teve pouca interferência nos resultados obtidos.

Como trabalhos futuros pretende-se analisar Round-Trip Time (RTT) e Throughput nos dois cenários propostos, assim como experimentar os testes de vazão em ambientes mais complexos.

REFERÊNCIAS

BOUCADAIR, M. et al. RFC 7426: **Rede Definida por Software (SDN): Camadas e Arquitetura Terminologia.** 2015. Disponível em: <https://tools.ietf.org/html/rfc7426>. Acesso em: 24 ago. 2019.

DE OLIVEIRA, Rodrigo A. et al. Analisando o Comportamento de Ambientes de Experimentação baseados em OpenFlow. In: Anais do XXII Workshop de Gerência e Operação de Redes e Serviços (WGRS-SBRC 2017). SBC, 2017.

DINIZ, Pedro Henrique; JUNIOR, Nilton Alves. Ferramenta IPERF: geração e medição de Tráfego TCP e UDP. Notas Técnicas, v. 4, n. 2, 2014.

FARIAS, Fernando NN et al. Implementação de um Novo Datapath OpenFlow em Ambientes de Switches Legados. 2011.

FEAMSTER, Nick; REXFORD, Jennifer; ZEGURA, Ellen. The road to SDN: an intellectual history of programmable networks. **ACM SIGCOMM Computer Communication Review**, v. 44, n. 2, p. 87-98, 2014.

FERNANDES, Eder Leao; ROTHENBERG, Christian Esteve. OpenFlow 1.3 software switch. **Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC**, p. 1021-1028, 2014.

GUEDES, Dorgival et al. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 30, n. 4, p. 160-210, 2012.

IRAWATI, Indrarini Dyah; NURUZZAMANIRRIDHA, Mohammad. Spanning Tree Protocol Simulation Based on Software Defined Network Using Mininet Emulator. In: **International Conference on Soft Computing, Intelligence Systems, and Information Technology**. Springer, Berlin, Heidelberg, 2015. p. 395-403.

Iperf. The TCP/UDP Bandwidth Measurement Tool. Disponível em: <https://iperf.fr/>. Acesso: 02 set. 2019.

JAIN, Sushant et al. B4: Experience with a globally-deployed software defined WAN. In: **ACM SIGCOMM Computer Communication Review**. ACM, 2013. p. 3-14.

KAUR, Karamjeet et al. Programmable firewall using software defined networking. In: **2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)**. IEEE, 2015. p. 2125-2129.

KJHA, Rakesh et al. OpenFlow technology: A journey of simulation tools. International Journal of Computer Network and Information Security, v. 6, n. 11, p. 49, 2014.

KHONDOKER, Rahamatullah et al. Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In: **2014 World Congress on Computer Applications and Information Systems (WCCAIS)**. IEEE, 2014. p. 1-7.

KREUTZ, Diego et al. Software-defined networking: A comprehensive survey. **arXiv preprint arXiv:1406.0440**, 2014.

LIMA, João Carlos et al. Análise da aplicação do Floodlight em um ambiente SDN. In: **Anais da IV Escola Regional de Informática do Piauí**. SBC, 2018. p. 208-213.

LINKLETTER. Using the POX SDN controller. Disponível em: <http://www.brianlinkletter.com/using-the-pox-sdn-controller/>. Acesso em: 01 set. 2019.

MATTOS, Diogo MF et al. OMNI: Uma ferramenta para gerenciamento autônomo de redes OpenFlow. **Salão de Ferramentas do XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC'12**, 2011.

MCKEOWN, Nick et al. OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, v. 38, n. 2, p. 69-74, 2008.

MONTEZELO, Gabriel Henrique. Práticas Laboratoriais em SDN usando Mininet e POX. 2017.

ONF. Software-Defined Networking: The New Norm for Networks. white paper. Disponível em: <https://www.opennetworking.org>. Acesso em: 17 jul. 2019.

RAO, Sridhar. SDN Series Part Four: Ryu, a Rich-Featured Open Source SDN Controller Supported by NTT Labs. Disponível em: <https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs/>. Acesso em: 24 ago. 2019.
ROTHENBERG, Christian Esteve et al. OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. **Cad. CPqD Tecnologia, Campinas**, v. 7, n. 1, p. 65-76, 2010.

ROWSHANRAD, Shiva et al. A survey on SDN, the future of networking. **Journal of Advanced Computer Science & Technology**, v. 3, n. 2, p. 232-248, 2014.

SATASIYA, Dhaval et al. Analysis of software defined network firewall (sdf). In: **2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)**. IEEE, 2016. p. 228-231.

SEZER, Sakir et al. Are we ready for SDN? Implementation challenges for software-defined networks. **IEEE Communications Magazine**, v. 51, n. 7, p. 36-43, 2013.

YEGANEH, Soheil Hassas; TOOTOONCHIAN, Amin; GANJALI, Yashar. On scalability of software-defined networking. **IEEE Communications Magazine**, v. 51, n. 2, p. 136-141, 2013.