

IMPLEMENTAÇÃO DE UMA CONTROLADORA EM REDES DEFINIDAS POR SOFTWARE

Dueren Fabiani*
Daniel Delfini Ribeiro †

2022

RESUMO

O conceito em Redes Definidas por Software (Software Defined Networks) e a arquitetura OpenFlow oferecem um caminho para a implementação de uma arquitetura de rede programável. Esse modelo propõe a separação do planos de controle e do planos de dados. Neste contexto, é discutida a arquitetura OpenFlow, que permite a criação de aplicações para Redes Definidas por Software. Por fim é apresentado o simulador de rede SDN, o Mininet, que implementa a interface OpenFlow em um cenário de simulação de rede contendo um controlador POX, um switch OpenFlow e seis máquinas sendo duas delas servidores e o restante clientes. POX é usado para desenvolvimento mais rápido e prototipagem de novas aplicações de rede. O objetivo principal foi conceituar e implementar um balanceador de carga em redes SDN.

Palavras-chave: Redes Definidas por Software. OpenFlow. SDN. POX

ABSTRACT

The concept of Software Defined Networks and the OpenFlow architecture offer a way to implement a programmable network architecture. This model proposes the separation of control planes and data planes. In this context, the OpenFlow architecture is discussed, which allows the creation of applications for Software Defined Networks. Finally, the SDN network simulator, Mininet, is presented, which implements the OpenFlow interface in a network simulation scenario containing a POX controller, an OpenFlow switch and six machines, two of which are servers and the rest are clients. POX is used for faster development and

*<duerenfabiani.pf0364@academico.ifsul.edu.br>

†<daniel.delfini@passofundo.ifsul.edu.br>

prototyping of new network applications. The main objective was to conceptualize and implement a load balancer in SDN networks.

Keywords: Software Defined Network. OpenFlow. SDN. POX

1 INTRODUÇÃO

É indiscutível que as redes de computadores vêm sendo empregadas em grande escala, uma vez que a maioria das atividades da sociedade dependem do acesso à internet. Atividades estas realizadas por meio de computadores, smartphones, tablets, smart TVs, além da crescente demanda trazida pela *Internet of Things* (IoT) e a mais recente *Internet of Everything* (IoE), novos modelos que vem ganhando grande relevância no setor (SANTOS, 2017). Ambas, abrangem uma crescente gama de equipamentos que se interconectam, geram e compartilham dados dos usuários, necessitando, portanto, de uma infraestrutura de rede cada vez mais abrangente.

A infraestrutura das redes é composta por combinações de circuitos que são responsáveis pelo processamento do tráfego de pacotes de dados, a qual é complementada por uma camada de software de controle que é encarregado pelo suporte de uma extensa lista de protocolos. No entanto, para a realização de qualquer alteração mais complexa nas configurações de controle de encaminhamento ou de administração dos processos que são próprios dos equipamentos, necessita-se da execução de rigorosos testes antes de agregá-los na rede corporativa, podendo resultar em um processo vagaroso e até mesmo custoso (SHERWOOD, 2010).

Conforme Guedes (2012), os profissionais da área de redes de computadores vêm tendo iniciativas no desenvolvimento e implementação de infraestrutura de redes com recursos de programação, trazendo melhorias como maior abrangência, escalabilidade e flexibilidade de acesso e de fluxo de dados. Uma das soluções que vem ganhando um crescente destaque é o conceito de *Software Defined Networks* onde vem despertado a atenção de empresas que desenvolvem soluções para redes, bem como por profissionais de redes de computadores e a comunidade acadêmica.

Por meio do protocolo OpenFlow, os dispositivos responsáveis pelo roteamento de pacotes dispõem de uma interface que permite que uma controladora SDN tenha acesso a uma tabela de encaminhamento (LINS, 2015). De acordo com Ramos (2013), a tabela de encaminhamento se dá através do protocolo OpenFlow em switches. Essa arquitetura consiste na separação entre o plano de controle, este presente na controladora e o plano de dados, presentes no switches. O plano de controle é responsável pelas decisões sobre os novos fluxos de dados entrantes, mantendo-se ativo com os switches OpenFlow através de um canal seguro. O plano de dados presentes nos switches OpenFlow, são baseados em uma tabela de fluxos composta por regras de encaminhamento, cada regra presente na tabela de fluxo consiste em diferentes ações associadas aos fluxos de dados. As regras presentes na tabela de fluxo são mantidas pela controladora, é ela quem orienta os comutadores (switches) quando e onde encaminhar os novos fluxos de dados. A identificação do fluxo se dá através dos cabeçalhos, onde constam informações das ações sobre os pacotes. Este processo garante ao administrador uma gestão facilitada e centralizada toda arquitetura.

Atualmente existem diversos controladores para gerenciamento das SDN's, dentre os quais podemos citar o OpenDaylight¹, o ONOS², o Ryu³ e o Pox⁴. O emulador Minine

foi utilizado para emular o ambiente do controlador SDN e assim monitorar o comportamento da rede.

Assim, este trabalho fundamentou-se a partir da análise de uma rede definida por software. Pretende-se, portanto, através do controlador POX e partir da implementação do balanceador de carga verificar o comportamento da rede.

Este artigo está organizado da seguinte maneira: A seção 2 apresenta as tecnologias utilizadas no desenvolvimento do trabalho. A seção 3 apresenta os trabalhos relacionados, dando a base de ideias no que se fundamentou o atual trabalho. A seção 4 apresenta os resultados obtidos a partir da implementação da SDN por meio do emulador Mininet. Por fim, a última seção contém as considerações finais apresentando sugestões de trabalhos futuros.

2 TECNOLOGIAS UTILIZADAS NO DESENVOLVIMENTO

Nesta seção é apresentado o embasamento teórico sobre o qual o trabalho se fundamenta, assim como as tecnologias utilizadas no atual trabalho. Desta forma, são abordados conceitos e características das redes definidas por software e o protocolo *OpenFlow*.

2.1 Redes Definidas por Software

Software Defined Networking (SDN) é uma arquitetura de rede emergente que promete simplificar o gerenciamento da rede melhorando a utilização dos recursos da rede e impulsionar a evolução e a inovação em redes tradicionais. As SDN introduzem a abstração nas camadas de rede, separando o plano de controle do plano de dados para uma entidade externa denominada controladora, como é exemplificado na Figura 1 (KAUR, 2014). Esta divisão separa os dispositivos de rede do gerenciamento e permite que ambos os planos evoluem de forma independente, fornecendo flexibilidade e melhor gerenciamento em comparação a arquiteturas de redes tradicionais.

É importante definir os termos: plano de controle e plano de dados. Conforme (PEREIRA, 2020) afirma, o plano de dados é responsável por encaminhar as mensagens e informações entre os nós da rede, ele não toma decisões por quais caminhos os dados devem ser encaminhados e não carrega implementações de protocolos, a “inteligência” está centralizada no plano de controle, onde é coordenado os recursos e gerenciado as políticas da rede, portanto, ele executa as ações determinadas orquestrando o plano de dados, como por exemplo, para qual caminho o pacote de dados deve seguir.

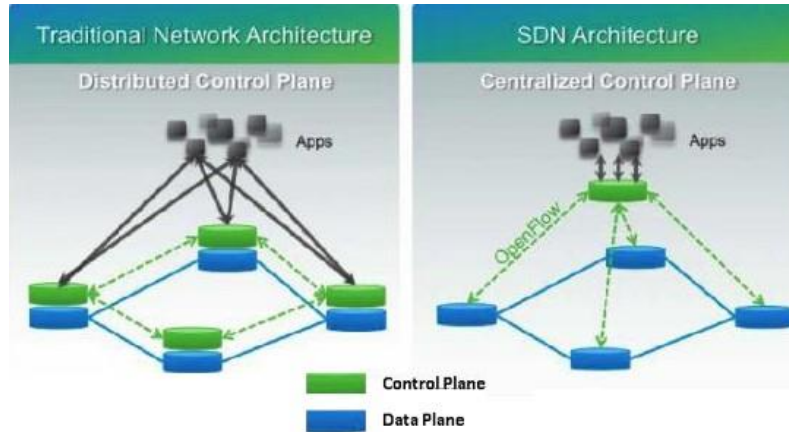
¹ <https://www.opendaylight.org/>

² <http://onosproject.org/>

³ <https://osrg.github.io/ryu/>

⁴ <https://github.com/noxrepo/pox>

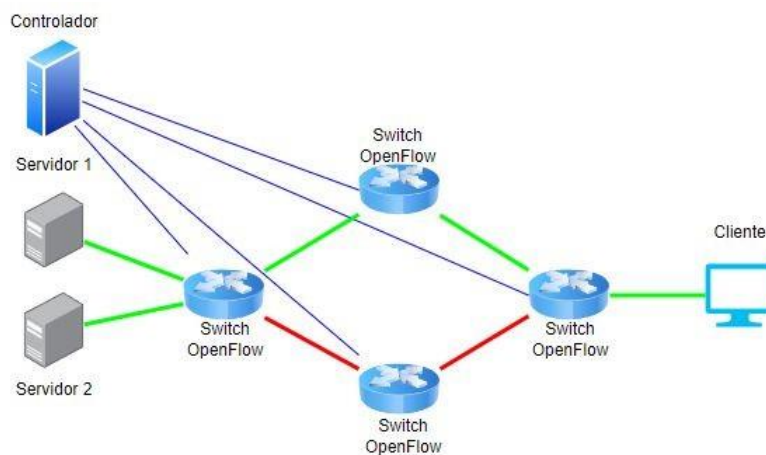
Figura 1 – Plano de dados e plano de controle



Fonte: (Lins, 2015)

A grande colaboração da arquitetura SDN é facilitar a rápida configuração e gerência da rede conforme a demanda de serviços (TEIXEIRA, 2018). A arquitetura SDN, pode ser uma ótima solução para melhorar a administração e gerenciamento da rede. Outro grande ponto positivo para o uso dessa arquitetura, é a automação de configurações e do monitoramento de erros, otimizando processos importantes. Como por exemplo, por algum problema um nó da rede passar a ser um gargalo, gerando lentidão ou perda de dados, conforme é ilustrado na Figura 2, a controladora que tem uma visão geral da rede irá calcular uma rota alternativa mais próxima para que o fluxo seja encaminhado.

Figura 2 – Mudança de fluxo com SDN



Fonte: Do autor, 2022

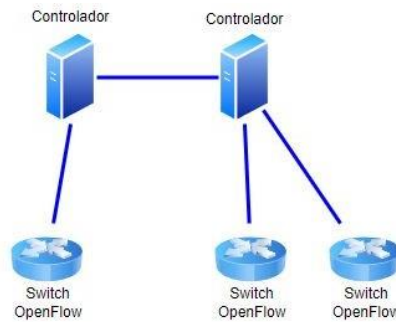
2.1.1 OpenFlow

Como bem nos assegura Barroso (2017) um dos protocolos vastamente estabelecido nas redes SDN é o OpenFlow, por ser um protocolo aberto, é possível definir tais regras em switches OpenFlow que tenham suporte, sejam programados por um controlador externo, definindo regras para cada tipo de fluxo, tendo essas regras armazenadas

em tabelas internas, e assim se utiliza destas tabelas para realizar os encaminhamentos do fluxo de dados.

Entre tantas características, um dos fatores mais relevantes no protocolo OpenFlow foi possibilitar que uma arquitetura distribuída possa ser aplicada ao controlador da rede, como ilustra a Figura 3, tornando uma rede centralizada, mas distribuída. Caso não houvesse essa possibilidade, a controladora poderia se tornar um gargalo, na medida em que o número do fluxo aumentasse (PANTUZA, 2019).

Figura 3 – Arquitetura distribuída



Fonte: Do autor, 2022

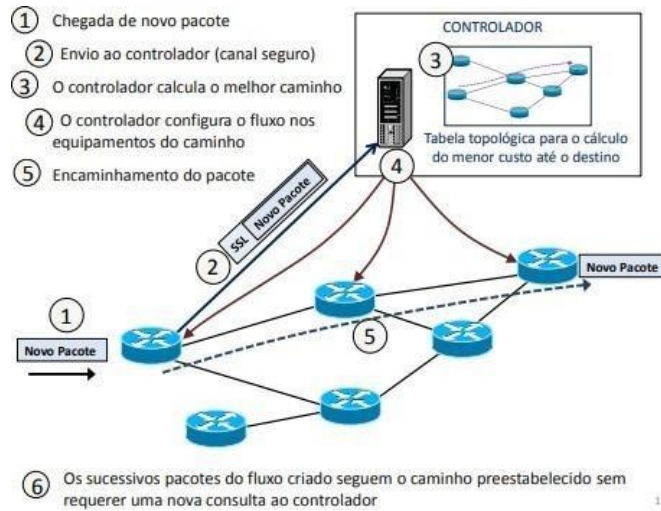
O mapeamento da topologia em uma rede SDN se dá através dos controladores que geralmente utilizam *Link Layer Discovery Protocol* (LLDP). Conforme Guedes (2012) afirma, o LLDP é um padrão aberto usado por ativos de rede para descobrir os vizinhos em redes. Em redes SDN, o controlador estrutura o frame LLDP e envia para todos os comutadores (switches). Quando um switch recebe o frame LLDP, ele encaminha esse frame para o controlador, e este consegue realizar o mapeamento da topologia da rede inteira.

Após construir uma visão total da topologia de rede, o controlador utiliza o *Spanning Tree Protocol* usada para prevenir *loops* durante *flooding* e *broadcast*. Diferente das redes tradicionais, o tráfego *unicast* não utiliza o *Spanning Tree Protocol*, ele seguirá o menor caminho entre a origem e o destino (HUANG, 2015).

Para cada fluxo de dados identificado, os switches OpenFlow podem executar pelo menos três ações: encaminhar o pacote conforme determinado na tabela de fluxos; encapsular e encaminhar o pacote para a controladora por meio de um canal seguro; descartar pacotes - usada para impedir os fluxos em determinada ocasião (RODRÍGUEZ, 2014).

Conforme é ilustrado na Figura 4, ao chegar o primeiro pacote de um novofluxo em um switch Openflow, o qual ainda não tem pré-configurado um fluxo no seu plano de dados para determinar seu encaminhamento, o pacote é encapsulado e enviado até o controlador por um canal seguro, o mesmo terá seu cabeçalho configurado e enviado para o comutador de origem (RODRÍGUEZ, 2014).

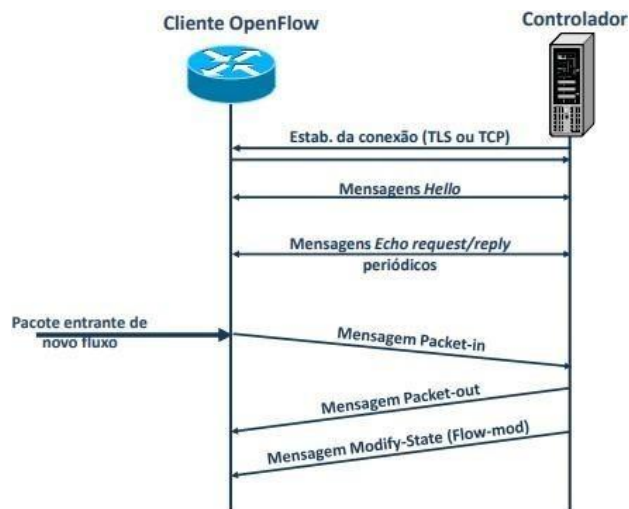
Figura 4 – Modo de funcionamento de uma SDN



Fonte: (Rodríguez, 2014)

Conforme é possível constatar a partir da Figura 5, é estabelecido através de um canal seguro uma conexão SSL (*Secure Socket Layer*) / TLS (*Transport Layer Security*) ou TCP é estabelecida caso o canal fique sem segurança. Em seguida o controlador e o switch OpenFlow passam a trocar as mensagens do tipo “hello” despertando a conexão. Após ser estabelecida a conexão ambos passam a trocar mensagens do tipo “echo” (*request/reply*). O novo pacote chega ao switch Openflow e é encaminhado para o controlador com uma mensagem *Packet-In*. Por fim, essa mensagem é respondida com um *Packet-out* estabelecendo como os comutadores devem se comportar ao encaminhar o pacote, configurando o fluxo específico no plano de dados de todos os dispositivos intermediários no trajeto, determinando o melhor e o menor caminho para o fluxo (RODRÍGUEZ, 2014). Segundo Torquato (2019), os próximos pacotes deste mesmo fluxo, quando já está pré-configurado o fluxo no plano de dados, não terão que realizar uma nova consulta à controladora.

Figura 5 – Caminho de um novo pacote



Fonte: (Rodríguez, 2014)

Os switches comerciais que suportam o protocolo OpenFlow, podem ser ativados através de uma atualização do *firmware* nos próprios equipamentos. Com essa atualização são adicionados aos switches a tabela de fluxo, o canal seguro e o protocolo OpenFlow. Vários fabricantes já oferecem produtos com a interface OpenFlow: Juniper, NEC, HP, Dell, OpenvSwitch, Cisco, Ciena, entre outros (LARA, 2014).

Conforme Ramos (2017) destaca também é possível utilizar do protocolo OpenFlow em máquinas virtuais ou em máquina física como hospedeira de uma controladora.

2.1.2 Tabela de Fluxos

Conforme Assis, (2019), a tabela de fluxos é um conjunto de regras e ações a serem tomadas. A regra é formada com base nos valores de um ou mais campos do cabeçalho do pacote, essa tabela consiste em um conjunto de entradas de fluxos como ilustrado na Figura 6.

De acordo com site oficial Open Networking Foundation (2014), uma tabela de fluxo consiste em:

- Match Field: Verifica se a entrada é aplicável ao pacote analisado, contém os dados da porta de entrada e cabeçalho dos pacotes.
- Priority: Define prioridade da entrada.
- Counter: Grava estatísticas dos pacotes corretos, quando são correspondentes à entrada.
- Instruction: Especifica as ações que devem ser tomadas nos pacotes.
- Counter: Grava estatísticas dos pacotes corretos, quando são correspondentes à entrada.
- Timeouts: Especifica o tempo de timeout do pacote, máximo que é considerado expirado pelo switch.
- Cookie: Usado pelo controlador para filtrar entradas de fluxo na tabela, filtrar estatísticas, modificações ou deleção de um fluxo.
- Flags: As flags alteram a maneira como as entradas de fluxo são gerenciadas.

Figura 6 – Campos de uma entrada na tabela de fluxo.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags
--------------	----------	----------	--------------	----------	--------	-------

Fonte: (Foundation, 2014)

2.1.3 Canal de Comunicação Seguro

Para que a rede não sofra ataques de indivíduos mal intencionados, o canal seguro garante confiabilidade na troca de mensagens entre o controlador e o switch OpenFlow.

Conforme afirma Assis (2019), é possível utilizar o protocolo SSL/TLS que utiliza a encriptação dos dados trafegados utilizando de certificados confiáveis. Entretanto, o canal seguro pode operar sem nenhuma configuração de segurança. A comunicação entre o controlador é feita através do protocolo OpenFlow, por diferentes tipos de mensagens, conforme é mostrado no site oficial Open Networking Foundation (FOUNDATION, 2014):

Mensagem simétricas: Podem ser geradas tanto pelo controlador quanto pelo switch.

Este tipo de mensagem é enviado sem solicitação de ambos, como exemplos:

- Hello - Quando uma conexão for estabelecida
- Echo - Identificação de latência, largura de banda e existência de conectividade
- Vendor - Provêem uma forma padrão para os switches OpenFlow oferecem funcionalidades adicionais

Mensagem assíncronas: Geradas pelo switch OpenFlow para atualizar o controlador sobre eventos da rede e mudanças no estado do switch.

- Packet-In - Entrada de pacotes, utilizado quando fluxos não classificados entram no switch
- Flow-Removed - Remoção de fluxo, mensagem enviada para o controlador, quando um fluxo é removido da tabela. Seja por *Idle* (Parada) *Timeout* (Tempo esgotado), *Hard Timeout* (Tempo limite difícil) ou por uma mensagem de modificação da tabela de fluxos que delete a entrada em questão
- Port-Status - Estado da porta, quando há mudanças nas configurações das portas
- Error - Erro, notificações de erros

Controlador para Switch - Geradas pelo controlador para gerenciar e o estado de um switch

- Features - O controlador requisita as características do switch OpenFlow
- Configuration – Configuração, usado para configurar ou solicitar configurações do switch OpenFlow
- Modify-State - Modificação de estado, usado para adicionar, deletar, modificar a tabela de fluxos, determinar propriedades nas portas do switch OpenFlow;
- Read-State - Coleta estatísticas
- Send-Packet - Usado para enviar pacotes por uma determinada porta do switch
- Barrier - Usado para garantir que as dependências foram atendidas ou para receber notificações de operações finalizadas

2.2 Controlador POX

POX é uma plataforma de desenvolvimento de código aberto para aplicativos de controle de rede definida por software (SDN) baseados em Python, como controladores SDN OpenFlow. O controlador utilizado neste trabalho é derivado do NOX clássico, um dos primeiros controladores OpenFlow. O POX apresenta como vantagens o fato de ser implementado em Python, uma linguagem de fácil entendimento, para o ambiente acadêmico. Além disso, o mesmo apresenta diversos componentes reusáveis e bibliotecas, sendo uma plataforma para o desenvolvimento e a prototipagem rápida de aplicações de *software* para SDNs (FERRARI, 2018).

2.3 Mininet

O Mininet é uma ferramenta para a simulação de Redes Definidas por Software que permite a rápida modelagem de uma infraestrutura virtual de rede com a utilização de apenas um computador. O mesmo também possibilita a criação de redes baseados em software como o OpenFlow, citado anteriormente, permitindo criar, interagir e customizar protótipos de Redes Definidas por Software de forma rápida (COSTA, 2013).

3 TRABALHOS RELACIONADOS

Nesta seção será apresentado um trabalho que faz uso da mesma tecnologia escolhida. Existem diversos outros trabalhos, porém este é o que mais se assemelha abordando os temas chave do presente trabalho.

Conforme Silva (2018) apresenta, em seu trabalho uma implementação de um firewall baseado em conceitos de SDN, utilizando um *Access Point* com protocolo OpenFlow e um controlador POX. O bloqueio de fluxos não autorizados foi feito a partir do endereço MAC (*Media Access Control*) dos hosts. Simulações foram conduzidas através do Mininet e um caso de uso foi implementado para testes reais. Os resultados obtidos em relação ao bloqueio de fluxos e latência da rede, mostram que é possível se definir, com eficácia e eficiência, funções de firewall através do SDN.

A diferença entre o trabalho pesquisado e este em desenvolvimento baseia-se em que, um foi implementado um balanceador de carga enquanto que o do autor que implementou um *firewall*, relacionando os mesmos conceitos baseados em SDN e o controlador POX. Fazendo o uso das mesmas tecnologias como Mininet, utilizando de um cenário semelhante, com 6 hosts diferentes, sendo um servidor Web e os outros como usuários comuns da rede, utilizando da SDN para definir o fluxo de acordo com as políticas de segurança propostas.

Tendo em vista o trabalho aprestado foi o que possibilitou na ideia do trabalho atual, nas próximas sessões será apresentada uma implementação prática de uma rede SDN.

4 IMPLEMENTAÇÃO DA SDN

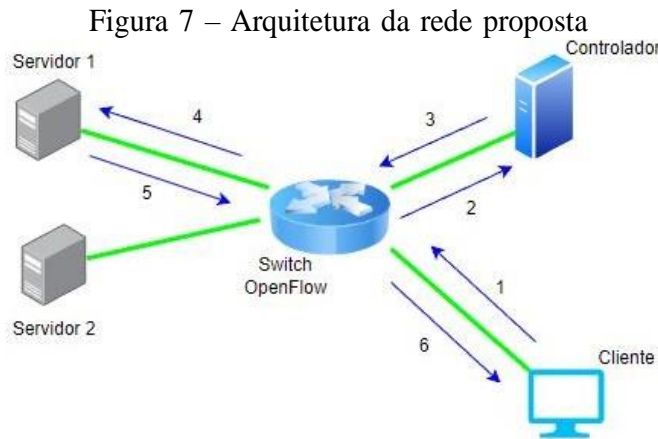
Com o objetivo de alcançar os resultados visados para o trabalho, foram utilizadas de algumas tecnologias como VirtualBox para emular o sistema operacional Ubuntu 21.10. No sistema foi instalado o Mininet, que se trata de um emulador de rede de computadores, possibilitando criar redes com servidores, switches, controladores e enlaces

virtuais, além destes o controlador POX que é plataforma para desenvolvimento rápido e prototipagem de controladores OpenFlow (*framework*) usando Python.

Nesta seção são apresentados mais detalhes sobre a implementação proposta, ferramentas utilizadas, bem como o detalhamento da implementação.

4.1 Arquitetura de Rede Proposta

A Figura 7 exemplifica a arquitetura da rede proposta: (1) onde é feita a requisição do cliente é enviada ao switch; (2) como não há entrada especificada na tabela de fluxos, o primeiro pacote da requisição é repassado ao controlador, para que o controlador tome uma decisão de encaminhamento do pacote; (3) o controlador escolhe o servidor de destino, cria uma entrada correspondente na tabela de fluxos e devolve o pacote ao switch; (4) a requisição é enviada ao servidor escolhido; (5) a requisição é atendida pelo servidor, que envia a resposta de volta ao switch; (6) a resposta da requisição é retornada ao cliente.



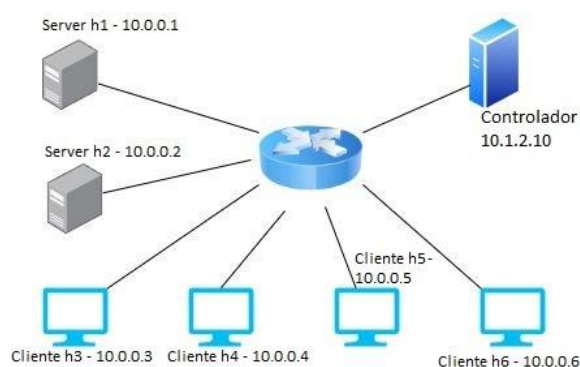
Fonte: Do autor, 2022

4.2 Características das Aplicações

Visando o objetivo de realizar uma implementação em SDN, buscou-se um cenário em que fosse possível a implementação dessa tecnologia. Tomando as iniciativas para realizar a implementação de um balanceador de carga em uma rede simulada. A seguir são apresentadas as funcionalidades da aplicação implementada. Para realizar a execução do presente trabalho, foi utilizado a linguagem de programação Python na versão 2.7.

A Figura 8 ilustra de forma mais clara como será estruturada a topologia de rede criada através do Mininet, com dois servidores WEB HTTP e quatro clientes, conectados a um switch onde está instalado o controlador SDN.

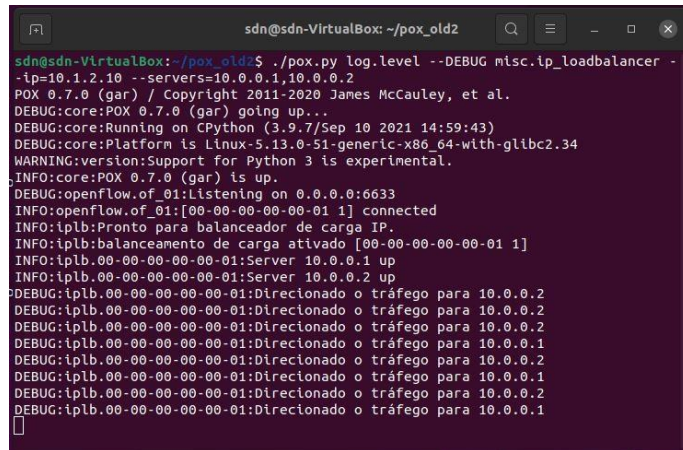
Figura 8 – Topologia



Fonte: Do autor, 2022

A Figura 9 apresenta o terminal do controlador aonde são mostradas algumas informações de recursos que nele são executados, como mostra: o momento em que o controlador se conecta ao switch OpenFlow, o estado do servidores em que ele realizará o balanceamento e o próprio direcionamento de tráfego para seus respectivos destinos.

Figura 9 – Distribuição de tráfego enviada para o host h1 e host h2



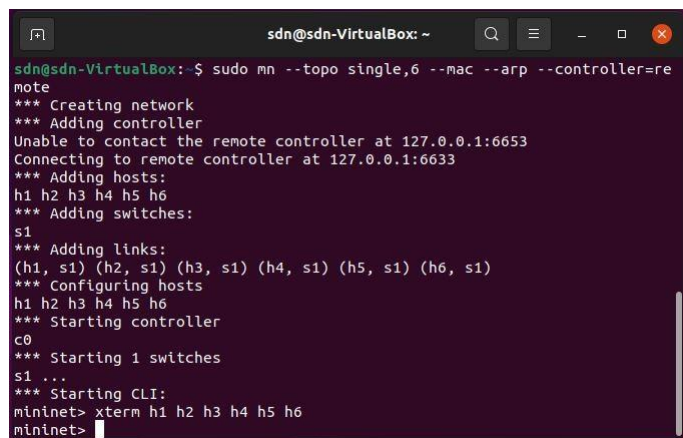
```
sdn@sdn-VirtualBox: ~/pox_old2
sdn@sdn-VirtualBox:~/pox_old2$ ./pox.py log.level --DEBUG misc.ip_loadbalancer -
-ip=10.1.2.10 --servers=10.0.0.1,10.0.0.2
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.9.7/Sep 10 2021 14:59:43)
DEBUG:core:Platform is linux-5.13.0-51-generic-x86_64-with-glibc2.34
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:iplb:Pronto para balanceador de carga IP.
INFO:iplb:balanceamento de carga ativado [00-00-00-00-00-01 1]
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up
INFO:iplb.00-00-00-00-00-01:Server 10.0.0.2 up
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.2
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.2
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.2
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.2
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.1
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.2
DEBUG:iplb.00-00-00-00-00-01:Direncionado o tráfego para 10.0.0.1
```

Fonte: Do autor, 2022

A topologia criada através do Mininet totalizando 6 hosts conectados a um Switch, como mostra a Figura 10, onde dois deles host h1, h2 são definidos como servidor WEB HTTP baseado em Python, conforme definido, estes servidores web rodam em endereços de IP 10.0.0.1 e 10.0.0.2, onde o controlador é definido no IP 10.1.2.10 e o balanceador de carga é executado conforme o comando a seguir:

```
sdn@sdn-VirtualBox# /pox_old2/pox.py log.level --DEBUG misc.ip_loadbalancer --
ip=10.1.2.10 --servers=10.0.0.1,10.0.0.2
```

Figura 10 – Rede simulada

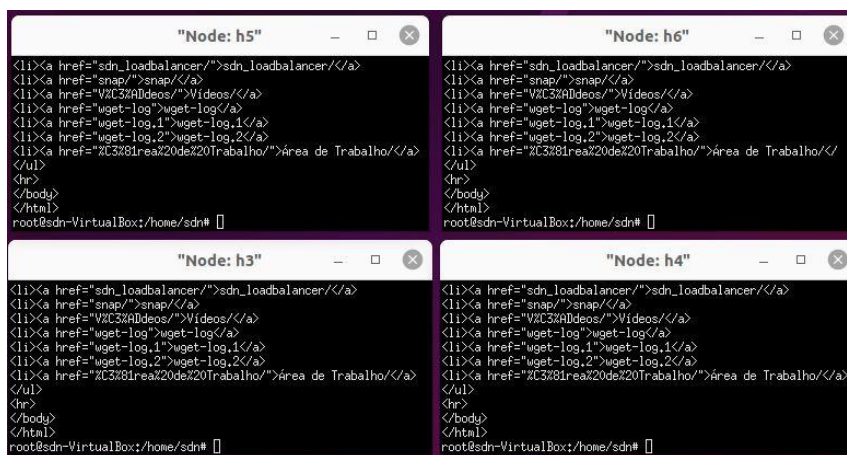


```
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox:~$ sudo mn --topo single,6 --mac --arp --controller=re
note
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2 h3 h4 h5 h6
mininet>
```

Fonte: Do autor, 2022

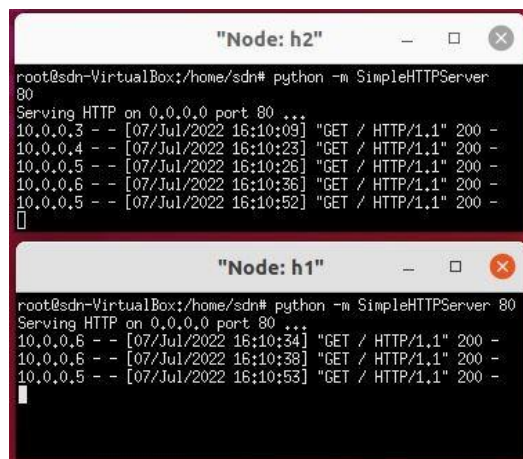
Como mostra na Figura 11, as solicitação WEB HTTP dos hosts h3, h4, h5, h6 é feita através do comando curl no IP 10.1.2.10, sendo nesse IP onde os clientes vão solicitar os dados para o respectivos servidores definidos na rede, onde a controladora está definindo como ira distribuir a carga para os host h1 ou h2, é onde está instado o serviço web, que estão rodando no endereço IP 10.0.0.1 e 10.0.0.2, bem como é mostrado na Figura 12.

Figura 11 – Terminal dos clientes



Fonte: Do autor, 2022

Figura 12 – Terminal dos servidores



Fonte: Do autor, 2022

No trecho de código abaixo da Figura 13, refere-se ao método *random*, o mesmo retornar um elemento aleatório de uma lista, neste caso uma lista de servidores, o método *choice()* retorna um elemento selecionado aleatoriamente da sequência especificada, essa sequência pode ser uma string, um intervalo, uma lista, uma tupla ou qualquer outro tipo de sequência, este trecho de código tem por objetivo retornando uma lista de servidores

para realizar o balanceamento de carga.

Figura 13 – Escolha do servidor

```
179 def _pick_server (self, key, inport):
180     """
181     Escolhe um servidor para uma nova conexão
182     """
183
184     return random.choice(list(self.live_servers.keys()))
185
```

Fonte: Do autor, 2022

Neste próximo trecho de código da Figura 14, refere-se função de direcionamento do tráfego, em sua situação principal onde não se tem o conhecimento do fluxo. Na linha 270 é definido de forma aleatória um servidor para redirecionar, executando a função da Figura 13, posteriormente é configurado a tabela de fluxo, definindo para onde o fluxo deve seguir e para qual porta deve ser encaminhado.

Figura 14 – Direcionamento do tráfego

```
258 elif ipp.dstip == self.service_ip:
259     # é para nosso IP de serviço e precisa ter balanceamento de carga
260     # se ele já conhecemos esse fluxo
261     key = ipp.srcip, ipp.dstip, tcp.srcport, tcp.dstport
262     entry = self.memory.get(key)
263     if entry is None or entry.server not in self.live_servers:
264         # Não conheço
265         if len(self.live_servers) == 0:
266             self.log.warn("Sem servidores!")
267             return drop()
268
269         # Escolha um servidor para este fluxo
270         server = self._pick_server(key, inport)
271         self.log.debug("Direcionado o tráfego para %s", server)
272         entry = MemoryEntry(server, packet, inport)
273         self.memory[entry.key1] = entry
274         self.memory[entry.key2] = entry
275
276     # atualizar timestamp
277     entry.refresh()
278
279     # configurar a entrada da tabela para o servidor selecionado
280     mac, port = self.live_servers[entry.server]
281
282     actions = []
283     actions.append(of.ofp_action_dl_addr.set_dst(mac))
284     actions.append(of.ofp_action_nw_addr.set_dst(entry.server))
285     actions.append(of.ofp_action_output(port = port))
286     match = of.ofp_match.from_packet(packet, inport)
287
288     msg = of.ofp_flow_mod(command=of.OFPFC_ADD,
289                           idle_timeout=FLOW_IDLE_TIMEOUT,
290                           hard_timeout=of.OFP_FLOW_PERMANENT,
291                           data=event.ofp,
292                           actions=actions,
293                           match=match)
294     self.con.send(msg)
```

Fonte: Do autor, 2022

CONSIDERAÇÕES FINAIS

Considerando que a implementação de redes SDN ainda é algo recente e que possui a tendência de crescimento, o trabalho teve como propósito implementar um script que controlasse uma rede de computadores, possibilitando que um software tome as decisões de tráfego de dados, realizando o balanceamento de carga entre dois servidores web.

Utilizando de conjunto de ferramentas para a implementação de uma Rede Definida por Software, possibilitando verificar o comportamento da rede, a maneira em que a controladora se comporta mediante às instruções definidas pelo administrador de forma centralizada. Sendo assim, foi possível observar que uma controladora SDN pode impactar em uma rede, pois as modificações realizadas como a implementação do balanceador de carga podem alterar a forma em que as informações fluem.

Com isso, para o desenvolvimento de um trabalho futuro, propõem-se a implementação de um algoritmo de escalonamento, como o Round-Robin (RR) sendo um algoritmos que permite o agendamento de processos em um sistema operacional, que atribui frações de tempo para cada processo em partes iguais e de forma circular, manipulando todos os processos sem prioridades, podendo ser aplicado para o escalonamento de requisições em redes SDN. Dessa forma, considera-se que os objetivos deste trabalho foram alcançados, permitindo-nos compreender sobre as Redes Definida por Software, e possibilitar como contributo para futuros trabalhos na área de redes, em especial em SDN.

REFERÊNCIAS

ASSIS, L. S. F. *OpenFlow*. 2019. Disponível em: <<https://www.gta.ufrj.br/ensino/eel879/vf/openflow/>>. Acesso em: 11 fev 2022. Citado 2 vezes nas páginas 7 e 8.

BARROSO, M. *Uma proposta de arquitetura de segurança para a detecção e reação a ameaças em redes SDN*. 2017. Disponível em: <<http://seer.upf.br/index.php/rbca/article/view/6595/4190/>>. Acesso em: 18 jan 2022. Citado na página 5.

COSTA, L. R. *OpenFlow e o Paradigma de Redes Definidas por Software*. 2013. Disponível em: <https://bdm.unb.br/bitstream/10483/5674/1/2013_LucasRodriguesCosta.pdf>. Acesso em: 02 jul 2022. Citado na página 10.

FERRARI, R. C. C. *Análise de tráfego entre OVS e controlador SDN*. 2018. Disponível em: <<https://proceedings.sbmac.org.br/sbmac/article/view/2090>>. Acesso em: 07 jul 2022. Citado na página 9.

FOUNDATION, O. N. *OpenFlow Switch Specification*. 2014. Disponível em: <<https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>>. Acesso em: 10 fev 2022. Citado na página 8.

GUEDES, D. *Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores*. 2012. Disponível em: <<https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/minicurso-sdn.pdf>>. Acesso em: 18 jan 2022. Citado 2 vezes nas páginas 2 e 6.

HUANG, T. *Path Computation Enhancement in SDN Networks*. 2015. Disponível em: <https://rshare.library.ryerson.ca/articles/thesis/Path_computation_enhancement_in_SDN_networks/14643900>. Acesso em: 26 jan 2022. Citado na página 6.

KAUR, S. *Network Programmability Using POX Controller*. 2014. Disponível em: <https://www.researchgate.net/profile/Japinder-Singh/publication/287216671_Network_Programmability_Using_POX_Controller/links/567447b508aebcdda0de21e6/Network-Programmability-Using-POX-Controller.pdf>. Acesso em: 01 jul 2022. Citado na página 3.

LARA, A. *Network innovation using openflow: A survey*. In: *Communications Surveys Tutorials*. 2014. Disponível em: <<https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1127&context=csearticles>>. Acesso em: 29 jan 2022. Citado na página 7.

LINS, T. *Redes Definidas Por Software (Software Defined Networks) SDN*. 2015. Disponível em: <<http://www2.decom.ufop.br/imobilis/redes-definidas-por-software-software-defined-networks-sdn/>>. Acesso em: 22 dez 2021. Citado 2 vezes nas páginas 2 e 4.

MININET. *Mininet*. Disponível em: <http://mininet.org/>>. Acesso em: 24 jan 2021.

PANTUZA, G. *O Protocolo OpenFlow*. 2019. Disponível em: <<https://blog.pantuza.com/artigos/o-protocolo-openflow>>. Acesso em: 19 jan 2022. Citado na página 5.

PEREIRA, J. *Um Plano de Controle Seguro e Distribuído para Redes Definidas por Software*. 2020. Disponível em: <https://repositorio.unb.br/bitstream/10482/39487/1/2020_JeffersonPereiradaSilva.pdf>. Acesso em: 23 dez 2021. Citado na página 4.

RAMOS, D. M. N. *Avaliação de desempenho de um controlador SDN implementado como um VNF*. 2017. Disponível em: <<https://repositorio.ufpe.br/bitstream/123456789/25355/1/DISSERTA%C3%87%C3%83O%20Danyel%20Mendes%20Nogueira%20Ramos.pdf>>. Acesso em: 15 jun 2022. Citado na página 7.

RAMOS, R. M. *Roteamento Resiliente em Redes de Data Center utilizando OpenFlow: Uma abordagem centrada em caminhos embarcados na origem*. 2013. Disponível em: <<https://intrig.dca.fee.unicamp.br/wp-content/papercite-data/pdf/ramos2013masterthesis.pdf>>. Acesso em: 20 jan 2022. Citado na página 2.

RODRÍGUEZ, F. L. *Arquitetura e protótipo de uma rede SDN-OpenFlow para provedor de serviço*. 2014. Disponível em: <<https://repositorio.unb.br/handle/10482/16030>>. Acesso em: 24 jan 2022. Citado 2 vezes nas páginas 6 e 7.

SANTOS, A. V. *Uma solução SDN para comunicação Mesh de Nós em uma rede Zigbee padrão 802.15.4*. 2017. Disponível em: <<http://www.repositoriobib.ufc.br/00001d/00001ddc.pdf>>. Acesso em: 08 jun 2022. Citado na página 2.

SHERWOOD, R. *Can the production network be the testbed?* 2010. Disponível em: <<http://dl.acm.org/citation.cfm?id=1924943.1924969>>. Acesso em: 27 jan 2022. Citado na página 2.

SILVA, L. *Avaliação, Através de Simulação e Teste Real, de um Firewall Baseado em SDN*. 2018. Disponível em: <https://www.academia.edu/25989471/Avalia%C3%A7%C3%A3o_Atrav%C3%A9s_de_Simula%C3%A7%C3%A3o_e_Testes_Reais_de_um_Firewall_Baseado_em_SDN>. Acesso em: 25 jul 2022. Citado na página 9.

SILVESTRIN, D. *Estudo comparativo sobre virtualização*. 2013. Disponível em: <<https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1127&context=csearticles>>. Acesso em: 21 jun 2022. Citado na página 10.

TEIXEIRA, E. V. *Virtualização (SDN E NFV) com ênfase em desempenho de rede*. 2018. Disponível em: <http://riut.utfpr.edu.br/jspui/bitstream/1/19367/1/CT_TELECOM_IV_2018_03.pdf>. Acesso em: 01 jul 2022. Citado na página 4.

TORQUATO, G. F. *Balaceamento de Crga em SDN Utilizando Tabelas de Grupo OpenFlow*. 2019. Disponível em: <<https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/minicurso-sdn.pdf>>. Acesso em: 27 dez 2021. Citado na página 7.