UMA ANÁLISE DA FERRAMENTA MININET PARA TESTE DE LIBERAÇÃO DE FLUXO

Carlos Alexandre Carvalho Tojeiro[1]; Marcos Antonio Bicudo[2]; Thiago José Lucas

Resumo
As redes de computadores crescem com o passar do tempo, incorporando cada vez mais recursos e, portanto, tornando-se mais complexas. Para facilitar o gerenciamento faz-se necessário a utilização de determinadas técnicas, muitas vezes não centralizadas. A SDN (Software Defined Network) surge como uma alternativa e uma forte tendência na centralização do gerenciamento das redes de computadores. Frente às necessidades de se prever o comportamento da estrutura física real, o objetivo geral deste estudo foi de realizar testes em laboratórios virtuais para verificar como a estrutura se comportaria. Para isto utilizou-se a ferramenta de testes Mininet que simula uma SDN. Ela trabalha com o protocolo aberto OpenFlow para auxiliar em simulações onde são criadas redes de diferentes topologias para que administradores, operadores, pesquisadores e estudiosos possam realizar testes diminuindo impactos quando implementados em uma estrutura real. Nos testes realizados percebeu-se que a ferramenta Mininet conseguiu realizar a liberação do fluxo desejado no switch, por meio de comandos enviados do Controlador remoto.
Palavras-chave: SDN; OpenFlow; Mininet; Controlador.

Abstract
Computer networks have grown over time, increasingly incorporating features and, therefore, becoming more complex. For ease of management to use it is necessary for certain techniques, often not centralized. The SDN (Software Defined Network) is an alternative and a strong tendency to centralize the management of computer networks. Meet the needs of predicting the behavior of the actual physical structure, the aim of this study was to perform tests on virtual labs to see how the structure would behave. For this we used the Mininet testing tool that simulates a SDN. She works with the open protocol OpenFlow to assist in simulations which are created networks of different topologies for administrators, operators, researchers and scholars can conduct tests decreasing impacts when implemented in a real structure. In tests it was noticed that the Mininet tool has achieved the release of the desired flow in the switch through sent from the remote controller commands.
Keywords: SDN, OpenFlow; Mininet; Controller.

Introdução

As redes de computadores, com o passar dos anos, começaram a consumir cada vez mais recursos e serviços. Este crescimento trouxe a necessidade de controle mais eficaz dos dados trafegados. Os diversos dispositivos de redes existentes, como switches, roteadores, etc., contêm novos aplicativos e estes fazem com que as redes

[1] Especialista em Segurança de Redes de Computadores pela Faculdade de Tecnologia de Ourinhos - FATEC. E-mail: carlos.tojeiro@fatecourinhos.edu.br.
[2] Especialista em Segurança de Redes de Computadores pela Faculdade de Tecnologia de Ourinhos - FATEC. E-mail: marcosbicudo@hotmail.com.

apresentem demandas mais complexas para gerenciar e controlar os dados. Devido a estes fatores, começaram a surgir diferentes protocolos proprietários oriundos de vários fabricantes de hardware, cada qual oferecendo suporte para controle de seus dispositivos (CSOMA, 2014).

Estes dispositivos implementados em soluções baseadas em software e hardware proprietários começaram a engessar a rede tornando-se "caixas pretas" (GELBERGER, 2013).

Diante deste fator, gerentes de redes se depararam com o fato de que os fornecedores de dispositivos de rede não atendiam suas necessidades, no que diz respeito à inovação e desenvolvimento. Caso houvesse a necessidade de testar um novo tipo de protocolo, seria necessário fazer um pedido ao fabricante e ficar na espera por longos processos e etapas de desenvolvimento, o que poderia demorar meses ou até anos (GELBERGER, 2013).

Com isto, gerentes tinham que substituir equipamentos, ficando dessa forma dependentes desses fornecedores, pois os protocolos de controle só funcionavam em um hardware do mesmo fabricante, problema que trouxe o encarecimento da estrutura de rede (CSOMA, 2014).

Em vista desse impasse, visando uma solução, surge na Universidade de Stanford, o conceito de SDN (Software Define Network) ou Redes Definidas por Softwares, que apresentou opções visando simplificar o trabalho dos administradores de rede. A ideia consiste na criação de um protocolo único chamado OpenFlow que, oferece uma interface de rede aberta a ser padronizada e controlada remotamente.

Dessa forma, é possível separar o plano de dados, pacotes que trafegam na rede, do plano de controle, que se caracteriza na parte inteligente do software passando a ser controlada por uma entidade externa, em um servidor chamado de Controller, que cuida do roteamento, da política de segurança, QoS (Quality of Service), da engenharia de tráfego etc., facilitando assim o gerenciamento dos administradores (CSOMA, 2014).

Diante desta proposta, estudiosos, pesquisadores e a comunidade científica iniciaram o desenvolvimento de novas arquiteturas de implementação do núcleo da rede capazes de atender a flexibilidade que estava faltando. Com o uso da programabilidade começaram a surgir novas APIs (Interface de Programação de Aplicativos), que fornecem um alto desempenho e controle da rede (GELBERGER, 2013).

Estudos destacam que os benefícios de SDN no que tange a possibilidade de centralizar o controle dos dispositivos da rede geram maior flexibilidade de

gerenciamento de fluxos de todos os hosts conectados em uma rede, facilitando a escalabilidade bem como capacidade de programação para aplicação das regras de controle (OLIVEIRA, 2014).

1 *OpenFlow*

A Rede Definida por *Software* facilita o controle da rede, gerenciando os dispositivos a partir de uma localidade central. Trabalhando em conjunto com *OpenFlow*, protocolo que repassa as regras de encaminhamento para entradas de tabelas de fluxos mantidas nos equipamentos (PAL, 2014).

A maior parte dos *switches* modernos utiliza-se de endereços de memória para que possam encaminhar os pacotes e implementar funcionalidades de segurança, prioridades de serviços e colher estatísticas. O protocolo *OpenFlow* é capaz de trabalhar com estes dados armazenados e ainda fazer-se passível de programação conforme a necessidade de cada rede (AMORIM, 2014).

No dispositivo tradicional, o encaminhamento de pacotes (*data path*) se dá no próprio equipamento, assim como as decisões de roteamento (*control path*). Em um *switch* capaz de trabalhar com protocolo *OpenFlow* o encaminhamento ocorre no dispositivo e as decisões de roteamento, passam para um Controlador que se comunica com o dispositivo por meio de mensagens enviadas, escolhendo rotas e regras previamente definidas por um administrador (AMORIM, 2014).

Ao utilizar o protocolo *OpenFlow*, o administrador conseguirá um maior controle de sua rede, uma vez que não necessita configurar regras de *firewall*, VLANs ou regras de controle de acesso de modo descentralizado, o que na rede convencional acontece, quando uma rede possui muitas *switches*/roteadores ou mais de um *firewall* (SHIE-YUAN, 2014).

Quando inicia-se um Controlador na rede, os dispositivos que trabalham com suporte ao protocolo *OpenFlow* começam a se comunicar com o Controlador por meio de um canal seguro chamado *OpenFlow Channel*, enviando mensagens, mesmo que não tenha sido adicionada qualquer regra na tabela do *Switch*/Roteador (AMORIM, 2014).

O canal seguro serve para garantir a confiabilidade na troca de mensagens entre o *switch* e o controlador, utilizando o protocolo SSL (Secure Socket Layer) (AMORIM, 2014).

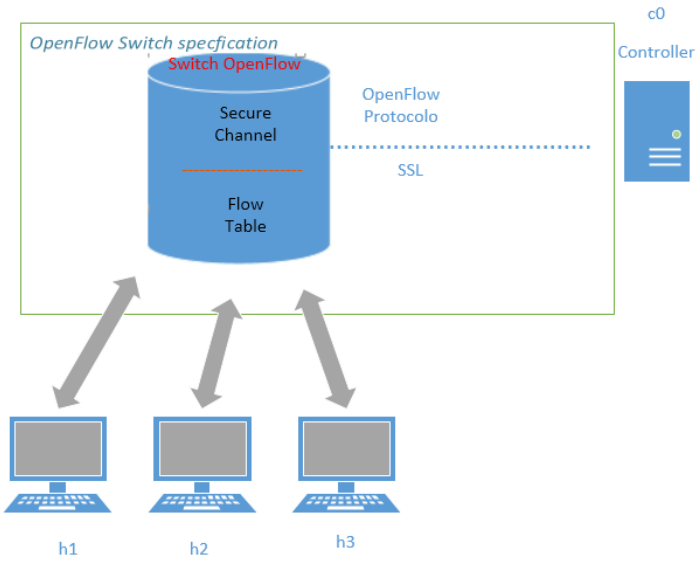

Figura 1 - *OpenFlow Secure Canal.*
Fonte: Adaptada de COSTA, 2014.

As mensagens *OpenFlow* enviadas podem ser do tipo:

- *Hello - Controller→Switch* (negociando a versão do protocolo).
- *Hello - Switch→Controller* (negociando a versão do protocolo).
- *Features Request - Controller →Switch* (solicitação do Controlador para saber quais portas estão disponíveis).
- *Set Config - Controller→Switch* (solicitação dos fluxos que foram expirados).
- *Features Reply - Switch→Controller* (*Switch* enviando ao controlador lista de portas, velocidade de portas e tabelas suportadas).
- Port Status - *Switch→Controller* (*Switch* enviando informações de velocidades e conectividades da portas ao Controlador).
- *Packet-In - Switch→Controller* (*Switch* enviando mensagem ao Controlador, informando que o pacote não possui entrada em sua tabela de fluxo).
- *Packet-Out - Controller →Switch* (Controlador envia um pacote de uma ou mais portas do *Switch*).
- *Flow-Mod - Controller →Switch* (Controlador adiciona uma entrada na tabela de fluxo do *Switch*).
- *Flow-Expired - Switch→ Controller* (*Switch* informa Controlador que o Fluxo expirou por inatividade).

Estas mensagens iniciam-se após o *three way handshake* do protocolo TCP (*Transmission Control Protocol).* O Controlador e o *switch* negociam a versão que será utilizada na comunicação e assim começa a troca de mensagens (MARCONDES, 2011).

Quando a *Switch* não sabe o que fazer com o pacote enviado, pergunta ao Controlador que ação deverá ser realizada (MARCONDES, 2011).

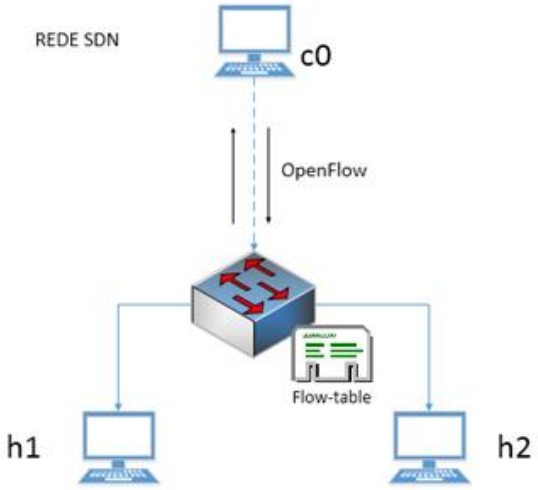

Figura 2 - *flow-table.*
Fonte: Elaborado pelos Autores.

Uma vez que o Controlador adiciona/remove a regra na *Flow-Table* o fluxo será liberado/bloqueado, assim o dispositivo encerra o encaminhamento para o Controlador, pois com os dados na tabela já se sabe a ação a realizar (MARCONDES, 2011).

Tabela 1 - Tabela de fluxos

Port	VLAN	ETH-SRC	ETH DST	IP-SRC	IP-DST	IP-PROTO	L4-SRC	L4-DST	AÇÕES
*	*	*	E0:06	*	*	*	*	*	Port 1
*	*	*	*	*	*	TCP	*	22	Drop
*	1	*	E0:06	*	*	*	*	*	Port 4,6,9

Fonte: Adaptado de Costa (2013).

Para melhor visualização dos objetos, deve ser previsto um espaço simples entre texto-objeto. Os Títulos de Tabelas e Quadros virão acima, em fonte tamanho 10. A seguir, apresenta-se um exemplo:

O campo " * " na tabela 1 representa que qualquer valor é aceito, não interferindo no reconhecimento de fluxo.

Percebe-se que o protocolo *OpenFlow* pode realizar ações simples, contudo, fornece muitas possibilidades de desenvolvimento por meio de combinações para aplicações em uma rede (MARCONDES, 2011).

2 Mininet

Dentro deste contexto de SDN e *OpenFlow* surge a ferramenta Mininet, utilizada em simulações de Redes Definidas por *Software*.

O Mininet é uma ferramenta que possibilita a criação de ambientes de testes, para que pesquisadores e estudiosos possam realizar experimentos de forma rápida em um computador contendo uma configuração básica (MARCONDES, 2011).

Oferecendo uma rede virtual escalável em sistema operacional utilizando o protocolo *OpenFlow*, permite a simulação de uma rede customizada antes de sua implantação (GELBERGER, 2013).

O Mininet é um sistema que simula a criação rápida de protótipos de uma rede em apenas um único computador, originando redes escaláveis definidas por *software*. Esses recursos permitem que a ferramenta Mininet venha interagir, personalizar e partilhar seus protótipos rapidamente (MARCONDES, 2011)

Capaz de emular vários *links*, *hosts*, *switches* e controladores *OpenFlow* em um único *Kernel*, o Mininet, dispõe de comandos para a realização de testes que são similares ao encontrado no terminal Linux. É possível também personalizar topologias utilizando a linguagem de programação *Python* (COSTA, 2013).

(i) ***Links*:** Um par *Ethernet* emulado representando duas interfaces conectadas;

(ii) ***Host*:** Dispositivo que representa uma instância da interface de rede independente;

(iii) ***Switches OpenFlow*:** Dispositivos virtuais que podem ser remotamente configurados e gerenciados pelo Controlador. A maioria dos *switches OpenFlow* inicia-se com uma porta de escuta passiva;

(iv) ***Controller*:** "Nó" que executa o controle por meio de um console podendo ser móvel ou não, usados para conexão com os dispositivos e para o encaminhamento das regras de entradas de fluxo dos *switches* e roteadores.

(v) **CLI:** *Prompt* onde são executados os comandos do Mininet.

No console fornecido pelo Mininet é possível controlar os dispositivos da rede emulada, por meio de uma CLI (*Command Line Interface*). O Mininet tem por objetivo, facilitar pesquisas nas redes definidas por *software* utilizando do protocolo *OpenFlow*, para desenvolvimento e pesquisas (KETI, 2015).

Alguns dos comandos possíveis na ferramenta Mininet são:

- **ping**→ utilizado para verificação de conexão entre dispositivos na rede.

- **ifconfig→** comando para verificar a configuração da interface de rede.
- **nodes→** exibir os nós da rede.
- **net→** exibir links.
- **dump→** exibir informações de todos os nós.
- **pingall→** faz com que todos os *hosts* enviem pings uns aos outros.
- **iperf→**para analisar a largura de banda entre dois *hosts* da rede.
- **dpctl→** permite visibilidade sobre a tabela de fluxo de um *switch*.
- **mn→** comando para criar a topologia desejada.
- **cbench →** para testar a taxa de instalação de fluxos (*flow setup rate*) em controladores *OpenFlow*.

Quanto a escalabilidade, a ferramenta Mininet permite emular até 4096 hosts em um único computador, porém perde desempenho quando os recursos solicitados em uma topologia criada ultrapassar os disponíveis na CPU que foi emulada ou exigir largura de banda superior a da máquina física (MISIC, 2014).

Por outro lado, Mininet é uma ferramenta confiável com desempenho rápido, que resulta em uma plataforma eficaz para que a comunidade SDN realize testes com alta fidelidade (YAN, 2015).

Percebe-se que a ferramenta Mininet fornece um modo simplificado, sem custos para que pesquisadores possam ser independentes ao realizar testes, mesmo em uma topologia complexa, sem a necessidade de uma estrutura física real, com ferramentas de depuração oferece condições para que administradores possam desenvolver programas em *Python* que se encaixem em suas arquiteturas de redes (CSOMA, 2014)

3 Criação da Topologia Proposta para Realização dos Testes

Pretendeu-se, portanto, utilizar neste trabalho, o s*oftware open source* Mininet, com sistema operacional Linux Ubuntu 14.04 LTS e o protocolo *OpenFlow* 1.0 com a finalidade de realizar testes bem como simular uma topologia de rede para coleta de dados com ferramentas já existentes no próprio *software* Mininet. Outras ferramentas foram instaladas para captura de pacotes, conexões remotas e edições de textos facilitando o trabalho.

Neste sentido, emulou-se na ferramenta Mininet uma topologia simples, onde cinco *hosts* interligam-se a um *switch* da camada dois, que está capacitado para

trabalhar com o protocolo *OpenFlow*. Este *switch* também é o responsável em mantê-los conectados.

Criou-se um Controlador remoto, de onde foram adicionadas regras de fluxo na *flow-table* do *switch*.

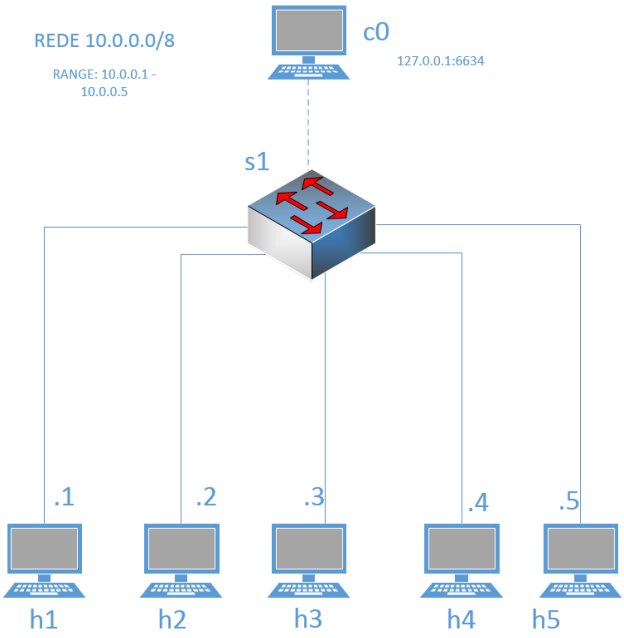

Figura 3 - Proposta de topologia simples para testes
Fonte**:** Elaborado pelos próprio Autores.

As regras enviadas do host 1 (h1) para o host 5 (h5) e do host 5 para o host 1, passaram antes pela ferramenta "dpctl" com o intuito de verificar os fluxos instalados no *switch* os quais liberam o tráfego. Em seguida realizou-se uma análise nos pacotes enviados do controlador para o *switch* verificando as mensagens *OpenFlow* entre o *switch* e o Controlador.

Dessa forma instalou-se a máquina virtual que contém a ferramenta Mininet, onde simulou-se uma rede SDN trabalhando com o protocolo *OpenFlow*.

Para iniciar a topologia na ferramenta Mininet utilizou-se o comando no terminal da máquina virtual:

mn –topo single,5 –mac –controller remote

No Mininet a rede virtual pode ser criada de duas maneiras: por meio de um *script Python* ou por linha de comando em um terminal. No exemplo, Fig.5, foi executado por meio de um terminal ssh (*Secure Shell*) utilizando-se da ferramenta *putty* instalada na máquina hospedeira.

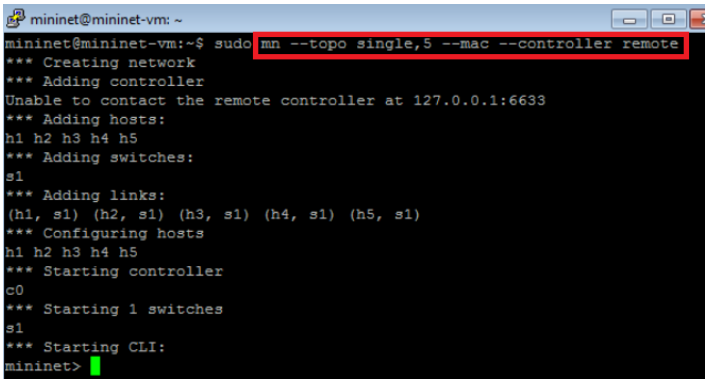

Figura 4 - Criação da rede proposta
Fonte: Elaborada pelos autores.

Na figura 5 é possível conferir a criação da rede, o *controller* (c0) , o *swtich* (s1) bem como os *hosts* (h1,h2,h3,h4 e h5) sendo inicializados.

Logo após a criação da estrutura de rede, verificou-se as informações sobre as conexões criadas pela ferramenta Mininet com o comando **links**

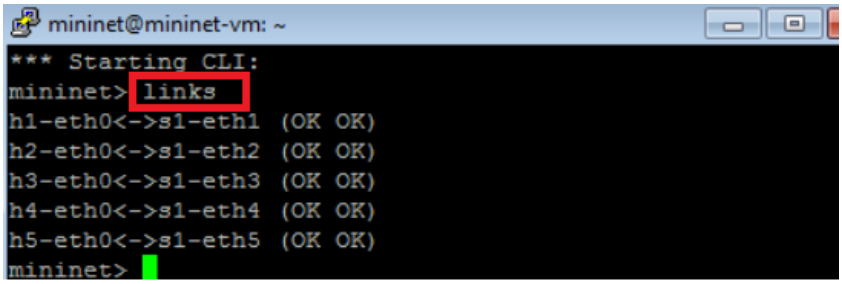

Figura 5 - Verificação dos *links*
Fonte: Elaborada pelos autores.

Conforme mostra a figura 5, é possível observar que o *host* 1 está conectado pela interface de rede eth0 com a *switch* 1 eth1(porta 1 do *switc*h s1), assim como os outros *hosts* sucessivamente na estrutura emulada no Mininet.

4 Preparação do Ambiente para Realização dos Testes

Utilizando-se de um servidor X (para iniciar ambiente de teste necessário) foram iniciadas três telas de terminais *host*1 (*Node* h1), *host*5 (*Node* h5) e um Controlador (Node c0) executando o comando na CLI da ferramenta Mininet: **xterm h1 h5 c0** →(abrir terminal h1, h3 e o *controller* c0)

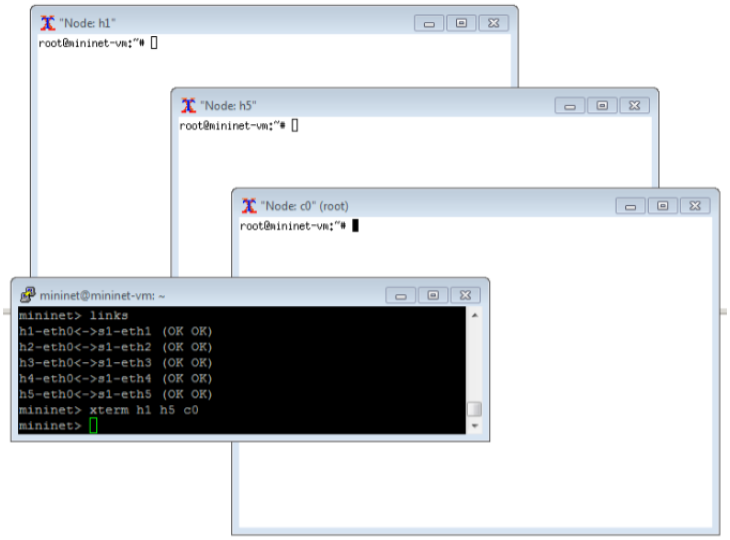

Figura 7 – Abrindo os terminais
Fonte: Elaborada pelos autores.

O Mininet, por padrão, inicia os *hosts* de rede com endereços IP a partir de 10.x.x.x/8, porém é possível definir outro endereço diferente a partir do arquivo de configuração (OLIVEIRA, 2014).

5. Realização dos Testes

Para realização dos testes utilizou-se a ferramenta de linha de comando "**dpctl**", capaz de enviar mensagens por meio do protocolo *OpenFlow* de um controlador para a *flow-table* no *switch*. O "**dpctl**" é aplicável para visualizar portas dos *switches*, estatísticas de fluxos e ainda fazer inserções de regras com um pacote FLOW_MOD encaminhado ao *switch*.

Em caso de erros no envio do pacote FLOW_MOD, caso o *switch* não reconheça as mensagens enviadas no pedido do controlador é importante salientar que será respondido com OF_FLOW_MOD_FAILED (PAL, 2014).

Com o comando: **dcpl dump-flows tcp:127.0.0.1:6634,** realiza-se um "**dump**" com uma conexão TCP. Na figura 8, é possível observar que está sendo realizado a partir do *controller* em uma tela do servidor X . Para conexão foi utilizado o IP (Protocolo de Internet) 127.0.0.1 interface local da máquina virtual. E a porta utilizada foi a 6634 padrão da ferramenta Mininet.

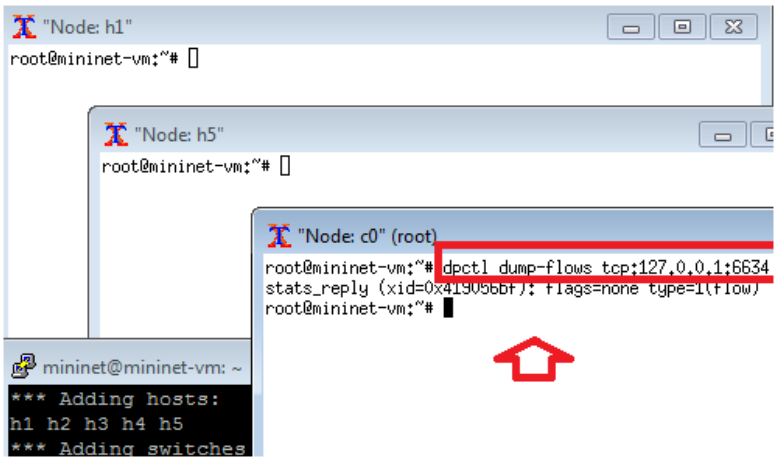

Figura 8 – Verificando a tabela de fluxo do switch s1
Fonte: Elaborada autores.

Desta forma, verifica-se na figura 8 que não existe fluxo entre os *hosts* ligados diretamente ao *switch*.

No terminal do "Nó" h1 foi disparado o comando "**ping"** de h1 (10.0.0.1) para h5 (10.0.0.5), onde obteve-se a resposta de "*host* de destino desconhecido". Verifica-se ainda que na imagem a tela do Nó h5 foi enviado comando "**ifconfig**" para certificar-se do endereço IP.

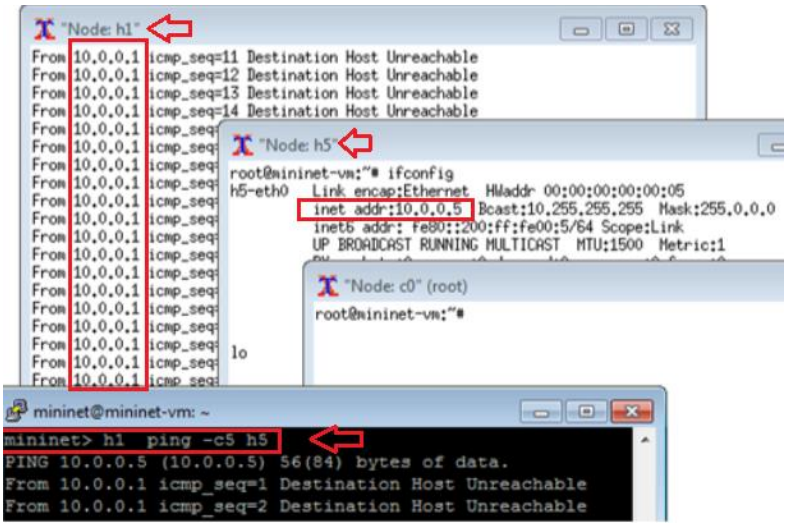

Figura 9 – Verificando o acesso de h1 para h5
Fonte**:** Elaborada pelos autores.

Prosseguindo com os testes, na figura 9, foi enviado o "**ping**" com cinco pacotes do *host*1 para o *host*5 diretamente no *prompt* da ferramenta Mininet, em conexão por ssh obtendo a mesma resposta "host de destino desconhecido".

No passo seguinte do teste proposto, foi enviado o comando para liberação do fluxo da porta de entrada 1(hum) para porta de saída 5(cinco) e outro da porta de entrada 5(cinco) para porta de saída 1(hum) sucessivamente.

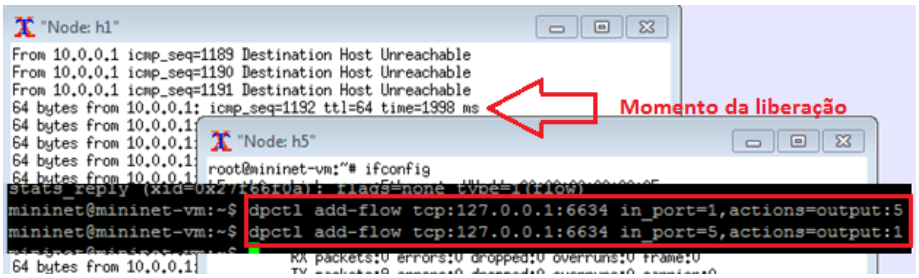

Figura 10 – Momento da liberação.
Fonte: Elaborada pelos autores.

Como é possível observar, na figura 10, utilizou-se o comando "**dpctl**" novamente conectando ao *controller* (c0) por meio da máquina virtual, adicionando a regra na *flow-table* do *switch* por meio do protocolo *OpenFlow*. Verifica-se ainda, o momento no qual o tráfego foi liberado.

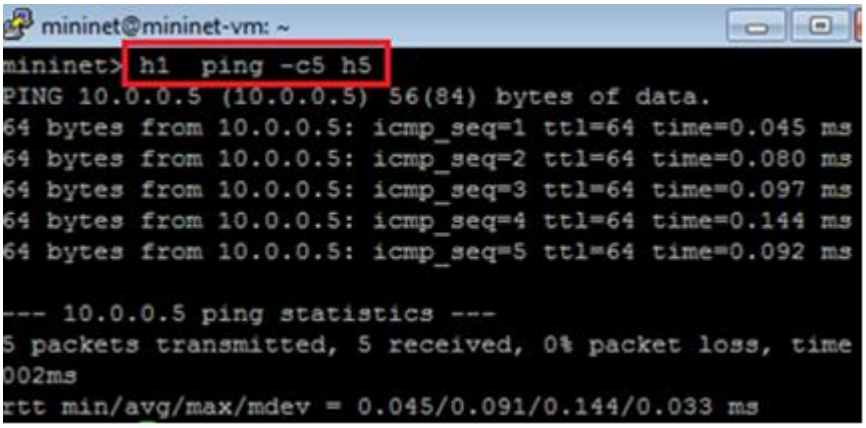

Figura 11 – Teste na ferramenta Mininet.
Fonte: Elaborada pelos autores.

Na figura 11 observa-se, que no *prompt* da ferramenta Mininet também foi liberado o fluxo após o envio das regras certificando-se do envio de cinco pacotes do *host* 1 para o *host* 5 com zero por cento de perdas.

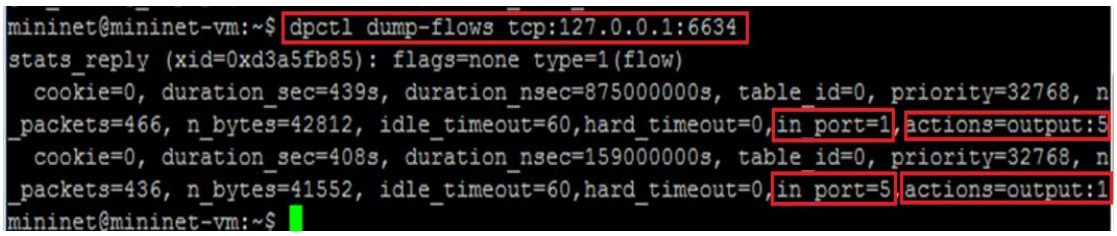

Figura 12 – Dump.
Fonte: Elaborada pelos autores.

Na figura 12 foi enviado um "**dump**" da máquina virtual pelo Controlador, com objetivo de verificar o fluxo passando pelo *switch*, onde é possível constatar o tráfego existente da porta de entrada 1 para porta de saída 5 e da porta de entrada 5 para porta de saída 1.

Nos testes também foi utilizado a ferramenta Wireshark de captura de pacotes, para que fossem verificadas as trocas de mensagens entre o Controlador e o *switch*. Em seguida, utilizou-se a ferramenta para analisar o pacote capturado por outra ferramenta o "**tcpdump**" ambas instaladas na máquina virtual.

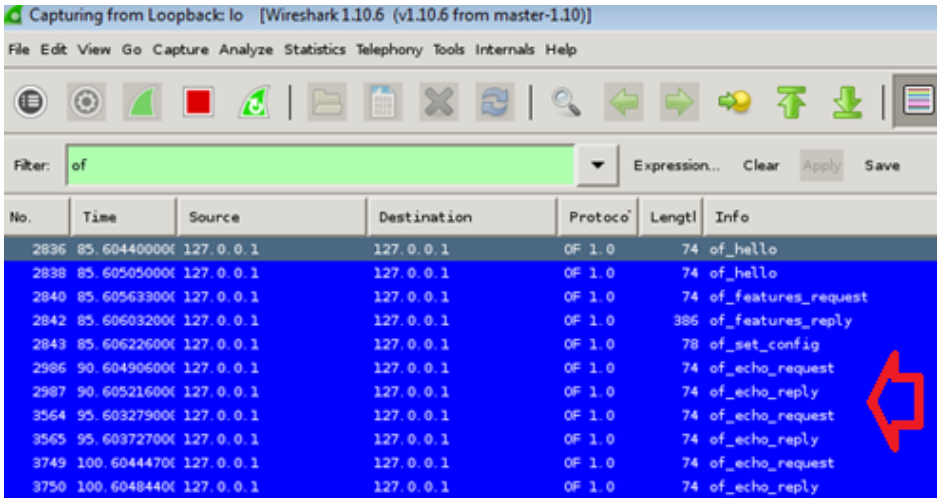

Figura 13 – Wireshark capturando pacotes *OpenFlow*.
Fonte: Elaborada pelos autores.

Conforme observa-se na figura 13, as mensagens foram capturadas (*hello*, *features request*, *set config*, *echo request*, *echo reply*) conforme citado no *OpenFlow*.

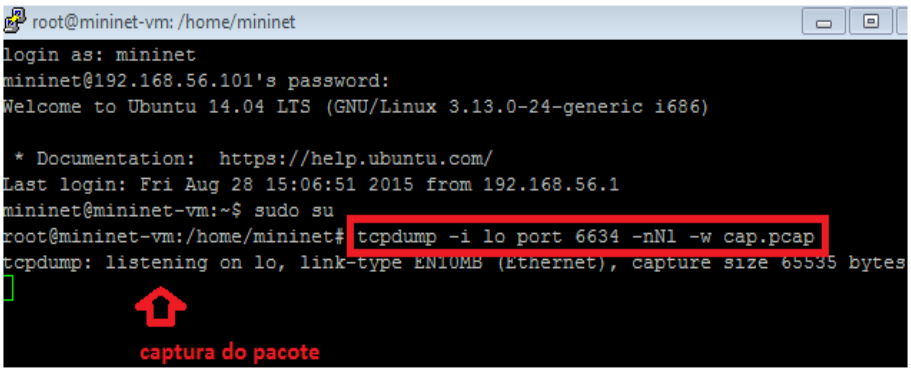

Figura 14 –Tcpdump.
Fonte : Elaborada pelos autores.

Na figura 14, visualiza-se a captura do pacote do Controlador para *switch* usando a ferramenta "**tcpdump**", este será direcionado para o arquivo para ser analisado-o com a ferramenta Wireshark. Pode-se também realizar esta captura diretamente no Wireshark que se encontra instalada na máquina virtual por meio do filtro "of&&(of.type !=5)&&(of.type !=1)".

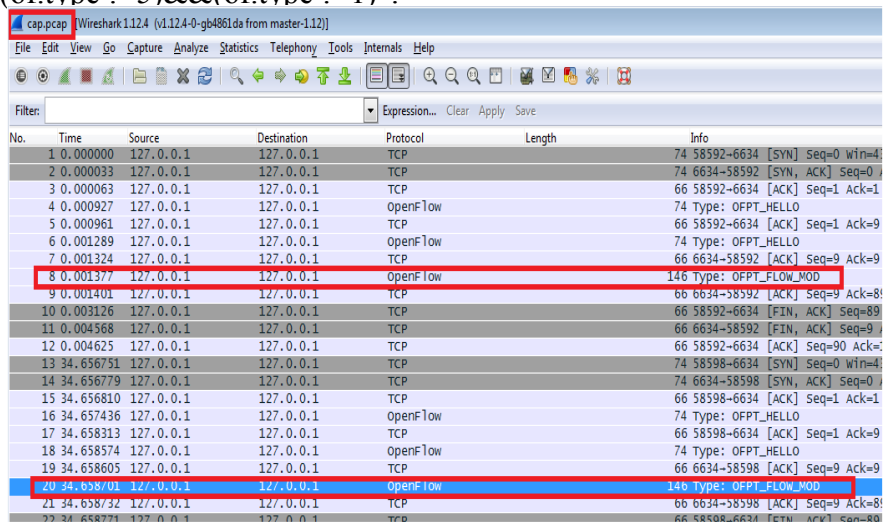

Figura 15 – Pacote OF_FLOW_MOD.

Fonte**:** Elaborada pelos autores.

Pode-se observar na figura 15 o pacote (*Flow-Mod*) com as mensagens enviadas do Controlador para *switch* , para que as entradas de fluxo sejam adicionadas na *flow-table*.

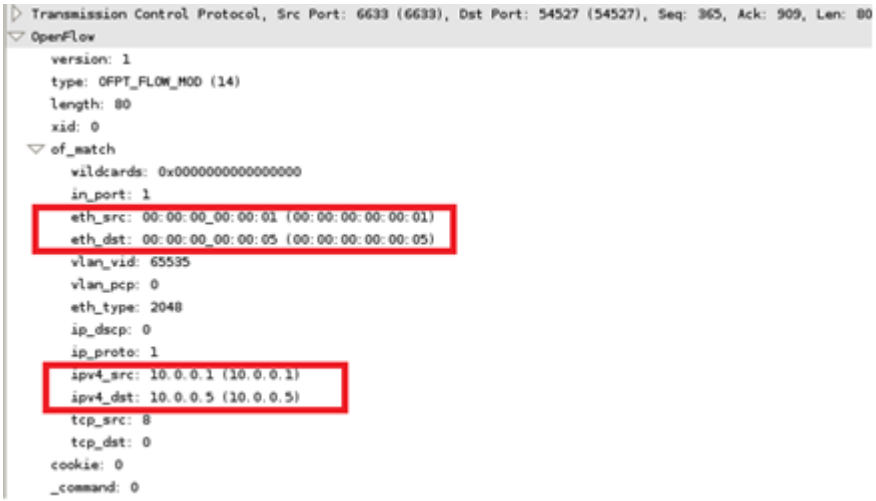

Figura 16 - Pacote OFPT_FLOW_MOD na porta 1 aberto.
Fonte: Elaborada pelos autores.

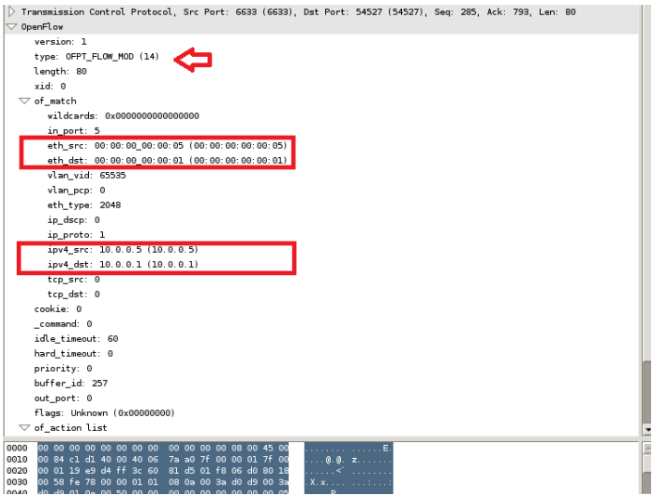

Figura 17 - Pacote OFPT_FLOW_MOD na porta 5 aberto.
Fonte: Elaborada pelos autores.

Constata-se que a tabela, agora está populada com as regras enviadas ao *switch* por meio do protocolo *OpenFlow*.

Podemos observar também o pacote OpenFlow Versão 1.0 tipo OFT_FLOW_MOD inseriu o fluxo na porta 5 e na porta 1 do switch com as regras envias pelo comando "dpctl" estabelecendo origem e destino com endereços IPs e MACs.

6. Conclusão

Nos testes realizados percebeu-se que a ferramenta Mininet, usada para simulações de Rede Defindа por *Software* conseguiu realizar a liberação do fluxo desejado no *switch*, por meio de comandos enviados do Controlador.

Verificou-se ainda a troca de mensagens (*Hello*, *Features Request*, *Features Reply* e *Set Config*), entre o Controlador e o *switch*, enviadas pelo protocolo *OpenFlow* por meio do *Secure Channel*, conforme é descrito na sua funcionalidade mesmo sem ter adicionado qualquer regra na tabela do switch. Conferiru *switch*.Conferiu-se o pacote Flow-Mod enviado ao *switch* para liberação do fluxo do *host*1 para o *host* 5 e do *host* 5 para *host* 1, sem erros, conforme ausência de mensagem de erro: OF_FLOW_MOD_FAILED (PAL, 2014).

Finalmente é possível concluir que, a priori, a ferramenta de testes Mininet apresentou-se de maneira confiável para simulações de uso do protocolo *OpenFlow* para o envio de liberação de fluxo. Contudo é recomendável ressaltar que os testes podem ser

feitos em ambientes reais verificando e comparando o desempenho com a ferramenta. Também com versões diferentes do protocolo *OpenFlow*.

Este trabalho não deve caracterizar-se em atividade fim. Pretende-se com ele, inspirar outras pesquisas e testes utilizando-se outras topologias, outras particularidades, mais máquinas, recursos e conectividade de *software*, visando melhorias a tudo o que tange ao gerenciamento de redes.

Referências

AMORIM, R. **O uso do protocolo Open Flow em redes definidas pro software.** 2014. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/1831/1/CT_GESER_II_2012_09.pdf>. Acesso em Junho, 2015.

COSTA, Lucas Rodrigues. **OpenFlow e o paradigma de redes definidas por software**. 2013. Disponível em: http://monografias.cic.unb.br/dspace/bitstream/123456789/391/1/Monografia_Vesao_Leitura_em_PC.pdf >. Acesso em Julho, 2015.

CSOMA, Attila, et al. **ESCAPE: Extensible service chain prototyping environment using mininet**, **click, netconf and pox**.2014.< http://www.researchgate.net/publication/265731807_ESCAPE_extensible_service_chain_prototyping_environment_using_mininet_click_NETCONF_and_POX>. Acesso em Agosto, 2015.

Gelberger, A.; Yemini, N.; Giladi, R., **Performance Analysis of Software-Defined Networking (SDN)**, in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on* , vol., no., pp.389-393, 14-16 Aug. 2013.

Keti, Faris, and Shavan Askar. **Emulation of Software Defined Networks Using Mininet in Different Simulation Environments.** *Proceedings of the 2015 6th International Conference on Intelligent Systems, Modelling and Simulation*. IEEE Computer Society, 2015.

Lantz, Bob, and Brian O'Connor. **A Mininet-based Virtual Testbed for Distributed SDN Development.** *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015.

Marcondes, Cesar. **Projeto de Desenvolvimento em OpenFlow**., *2011*.

Misic, M.J.; Gajin, S.R., "Simulation of Software Defined Networks in Mininet environment," in *Telecommunications Forum Telfor (TELFOR), 2014 22nd* , vol., no., pp.1055-1058, 25-27 Nov. 2014 doi: 10.1109/TELFOR.2014.7034588

Oliveira, R.L.S.de; Shinoda, A.A.; Schweitzer, C.M.; Rodrigues, Prete, L., **Using Mininet for emulation and prototyping Software-Defined Networks**, *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on* , vol., no., pp.1,6, 4-6 June 2014.

Pal, C.; Veena, S.; Rustagi, R.P.; Murthy, K.N.B., **Implementation of simplified custom topology framework in Mininet,** *Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on* , vol., no., pp.48,53, 10-12 Feb. 2014.

Shie-Yuan Wang, **Comparison of SDN OpenFlow network simulator and emulators**: EstiNet vs. Mininet, in *Computers and Communication (ISCC), 2014 IEEE Symposium on* , vol., no., pp.1-6, 23-26 June 2014.

Yan, Jiaqi, and Dong Jin. **VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation.** *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015.