Prototipação de Redes Definidas por *Software* (SDN) OpenFlow com Open vSwitch, Floodlight e VirtualBox

Rafael Fernando Diorio[1], Edivaldo Serafim[2], Karlan Ricomini Alves[3], Matheus Carvalho Meira[4]

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP. E-mail: rafael.diorio@ifsp.edu.br
[2]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP. E-mail: eserafim@ifsp.edu.br
[3]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP. E-mail: karlan.ricomini@ifsp.edu.br
[4]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – IFSP. E-mail: meira@ifsp.edu.br

Resumo: Com ampla aceitação pela academia e pela indústria, as Redes Definidas por *Software (Software-Defined Network, SDN)* tornaram-se uma das mais importantes arquiteturas de rede da atualidade. Por esse motivo, são constantemente abordadas em uma série de discussões e contribuições recentes no âmbito das redes de computadores, tais como relacionadas com questões acerca do gerenciamento, da segurança e da mobilidade, dentre outros. Diante desse cenário, este trabalho discorre sobre a prototipação de SDNs por meio de soluções de *software* específicas, nesse caso, embasadas no *software switch* Open vSwitch, no controlador SDN Floodlight, na solução de virtualização Oracle VM VirtualBox e por meio do protocolo OpenFlow. Para tal, um ambiente computacional configurado sobre o sistema operacional Linux Ubuntu é utilizado para fins de prototipação e experimentação de um cenário SDN OpenFlow de referência. Em linhas gerais, as discussões são realizadas de modo a possibilitar ao leitor a prototipação de SDNs OpenFlow por meio de soluções de *software* amplamente utilizadas no âmbito das redes de computadores, as quais podem ser empregadas, por exemplo, em atividades de ensino e/ou de pesquisa relacionadas ao tema em questão. Essa discussão é importante, por exemplo, para conciliar teoria e prática em disciplinas específicas dos cursos da área de informática do IFSP (e/ou de outras instituições de ensino), bem como para contribuir com outras discussões e abordagens no âmbito das SDNs, de modo geral.

Palavras–chave: Ensino. Experimentação. OpenFlow. Prototipação. Redes Definidas por *Software*.

Linha Temática: Ensino e Aprendizagem.

1 INTRODUÇÃO

Nos últimos anos, as Redes Definidas por *Software (Software-Defined Network, SDN)* tornaram-se uma das mais importantes arquiteturas de rede da atualidade. Isso se deve, em especial, por suas características arquiteturais e funcionais, tais como por meio da separação dos planos de dados e de controle e por meio de recursos que objetivam simplificar e flexibilizar o gerenciamento da rede, além de permitir e facilitar sua evolução e inovação (KREUTZ et al., 2015; XIA et al., 2015; NUNES et al., 2014), os quais são amplamente explorados por meio do protocolo OpenFlow (MCKEOWN et al., 2008). Como resultado, uma série de esforços da academia e da indústria discorre acerca de abordagens no âmbito das SDN, tais como em publicações recentes relacionadas ao tema (SHIRALI-SHAHREZA; GANJALI, 2018; BI et al., 2018; YAP et al., 2017; DIORIO; TIMÓTEO, 2016), bem como em soluções de mercado em SDN de empresas como Cisco[1] e Juniper[2], dentre outras.

Diante desse cenário, a prototipação de SDNs é importante para muitos propósitos, tais como para possibilitar a experimentação e avaliação de seus recursos arquiteturais e funcionais, para a realização de atividades de ensino e/ou de pesquisa relacionadas ao tema, bem como para novas contribuições no âmbito das redes de computadores, de modo geral. Com base em tais propósitos e objetivando contribuir com outros trabalhos relacionados ao tema (FONTES et al., 2015; DE OLIVEIRA et al., 2014; GUPTA; SOMMERS; BARFORD, 2013), este trabalho discorre acerca da prototipação de SDNs a partir de um ambiente computacional fornecido por meio do sistema operacional Linux Ubuntu e embasado nas soluções de *software* Open vSwitch, Floodlight, Oracle VM VirtualBox e no protocolo OpenFlow, ambos experimentados e discutidos a partir de um cenário SDN OpenFlow de referência. Essa discussão é importante, por exemplo, para conciliar teoria e

[1]Soluções SDN Cisco em https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html
[2]Soluções SDN Juniper em https://www.juniper.net/us/en/products-services/sdn/?

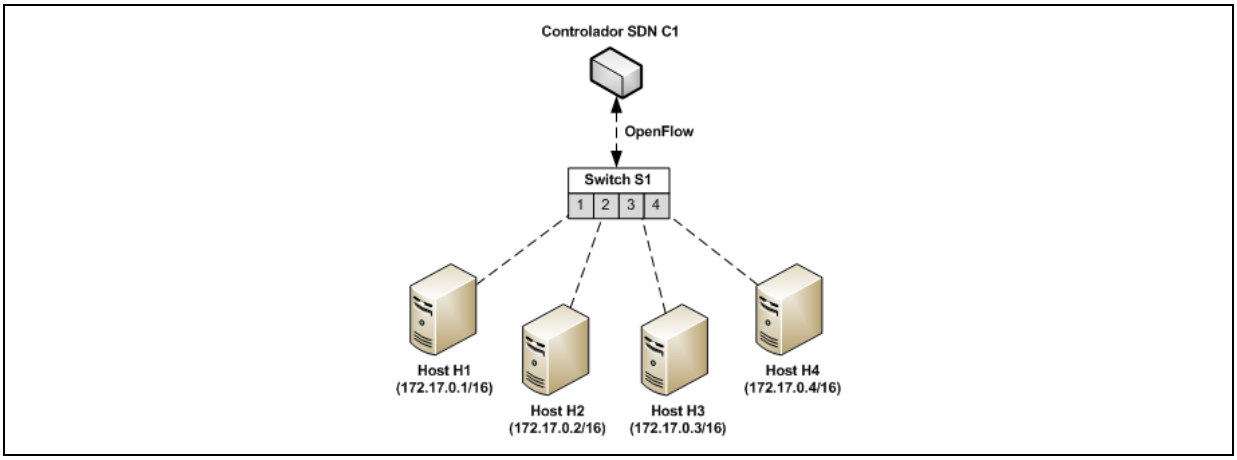
prática em disciplinas específicas dos cursos da área de informática do IFSP (e/ou de outras instituições de ensino), bem como para contribuir com outras discussões e abordagens no âmbito das SDNs, de modo geral.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta o cenário de referência para a prototipação e experimentação SDN OpenFlow empregado neste trabalho. A Seção 3 discorre sobre os materiais e métodos. A Seção 4 discorre sobre os resultados experimentais e, por fim, a Seção 5 apresenta a conclusão e os trabalhos futuros no âmbito deste trabalho.

2 CENÁRIO DE REFERÊNCIA

O cenário de referência para a prototipação e experimentação SDN OpenFlow empregado neste trabalho é ilustrado na Figura 1.

Figura 1. Cenário de referência para a prototipação e experimentação SDN OpenFlow empregado neste trabalho: Controlador SDN C1, *switch* S1 e *hosts* H1, H2, H3 e H4.



Nesse contexto, conforme ilustrado na Figura 1, o cenário de referência para a prototipação e experimentação SDN OpenFlow empregado neste trabalho é composto por um com controlador SDN (controlador C1), um *switch* SDN (*switch* S1) e quatro *hosts* (*hosts* H1, H2, H3 e H4). O *switch* S1 possui quatro interfaces de rede, em que a interface 1 é utilizada para a interconexão do *host* H1, a interface 2 é utilizada para a interconexão do *host* H2, a interface 3 é utilizada para a interconexão do *host* H3 e a interface 4 é utilizada para a interconexão do *host* H4. Nesse cenário, o *host* H1 possui endereço IP 172.17.0.1/16, o *host* H2 possui endereço IP 172.17.0.2/16, o *host* H3 possui endereço IP 172.17.0.3/16 e o *host* H4 possui endereço IP 172.17.0.4/16. Por sua vez, todo processo de comunicação entre o controlador C1 e o *switch* S1 é realizado por meio do protocolo OpenFlow.

3 MATERIAIS E MÉTODOS

Para a prototipação e experimentação do cenário SDN OpenFlow ilustrado na Seção anterior (Figura 1), as seguintes soluções de *software* foram empregadas neste trabalho: Oracle VM VirtualBox[3] versão 5.2.4, Open vSwitch[4] versão 2.5.4 e Floodlight[5] versão 1.2, ambas instaladas sobre o sistema operacional Linux Ubuntu[6] 16.04 LTS com suporte ao protocolo OpenFlow. Nesse cenário, a solução Oracle VM VirtualBox foi empregada para a implementação dos *hosts* do plano de dados da SDN (*hosts* H1, H2, H3 e H4), a solução Open vSwitch foi empregada para a implementação do *software switch* da SDN (*switch* S1) e a solução Floodlight foi empregada para a implementação do controlador da SDN (controlador C1).

Nesse contexto, após a instalação de tais soluções de *software* no Linux Ubuntu, algumas configurações foram necessárias para viabilizar a configuração de ambas conforme o cenário de

[3]Oracle VM VirtualBox em https://www.virtualbox.org
[4]Open vSwitch em http://openvswitch.org
[5]Floodlight em http://www.projectfloodlight.org/floodlight
[6]Linux Ubuntu em https://www.ubuntu.com

referência ilustrado na Figura 1. Dessa forma, a Figura 2 ilustra os comandos empregados para viabilizar a configuração do Linux Ubuntu e do Open vSwitch pertinentes ao *switch* S1 e suas respectivas interfaces de comunicação com os *hosts* da rede. De modo complementar, a Figura 3 ilustra os comandos empregados para viabilizar a comunicação do *switch* S1 com o controlador SDN C1 (Floodlight) e, por sua vez, a Figura 4 ilustra os comandos empregados para inicialização do controlador SDN C1 no cenário SDN OpenFlow de referência empregado neste trabalho.

Figura 2. Comandos empregados para viabilizar a configuração do Linux Ubuntu e do Open vSwitch pertinentes ao *switch* S1 no cenário SDN OpenFlow de referência empregado neste trabalho.

```
sudo ip tuntap add mode tap s1-1
sudo ip tuntap add mode tap s1-2
sudo ip tuntap add mode tap s1-3
sudo ip tuntap add mode tap s1-4
sudo ip link set s1-1 up
sudo ip link set s1-2 up
sudo ip link set s1-3 up
sudo ip link set s1-4 up
sudo ovs-vsctl add-br s1
sudo ovs-vsctl set bridge s1 protocols=OpenFlow10,OpenFlow13
sudo ovs-vsctl add-port s1 s1-1
sudo ovs-vsctl add-port s1 s1-2
sudo ovs-vsctl add-port s1 s1-3
sudo ovs-vsctl add-port s1 s1-4
```

Figura 3. Comandos empregados para viabilizar a configuração de comunicação do *switch* S1 com o controlador SDN C1 (Floodlight) no cenário SDN OpenFlow de referência empregado neste trabalho.

```
sudo ovs-vsctl set-controller s1 tcp:127.0.0.1:6653
sudo ovs-vsctl set-fail-mode s1 secure
```

Figura 4. Comandos empregados para viabilizar a inicialização do controlador SDN C1 no cenário SDN OpenFlow de referência empregado neste trabalho.

```
cd /home/user/floodlight/
sudo java -jar target/floodlight.jar
```

Em tal cenário, de modo complementar às configurações do *switch* S1 e do controlador C1, foi necessário associar as interfaces de rede dos *hosts* H1, H2, H3 e H4 da SDN OpenFlow com suas respectivas interfaces de comunicação junto ao *switch* S1, tal como ilustrado na Figura 5 para os *hosts* H1 e H2 e na Figura 6 para os *hosts* H3 e H4.

Figura 5. Configurações realizadas nas interfaces de rede dos *hosts* H1 (à esquerda) e H2 (à direita) para viabilizar suas interconexões com o *switch* S1 no cenário SDN OpenFlow de referência empregado neste trabalho.

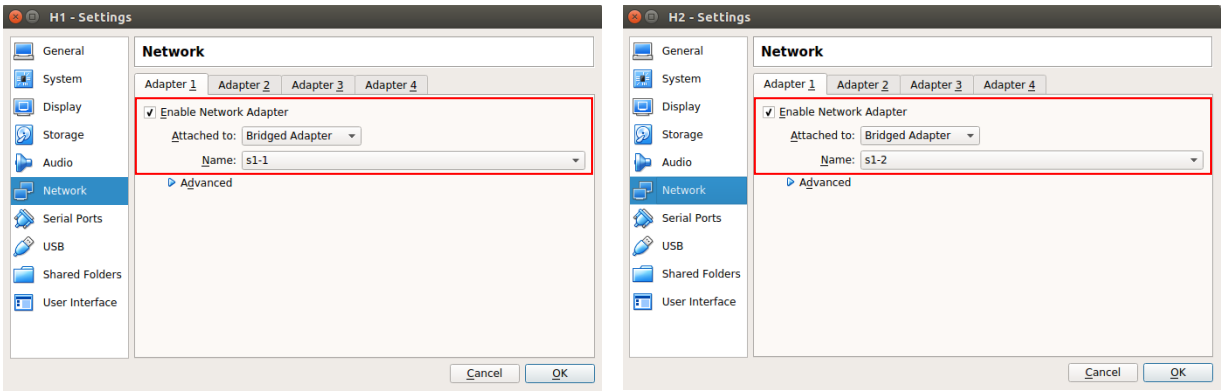
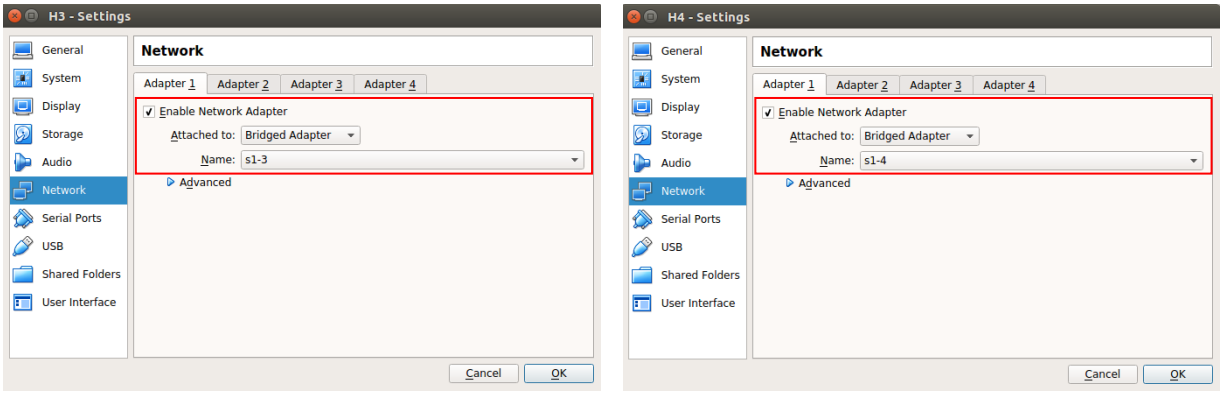


Figura 6. Configurações realizadas nas interfaces de rede dos *hosts* H3 (à esquerda) e H4 (à direita) para viabilizar suas interconexões com o *switch* S1 no cenário SDN OpenFlow de referência empregado neste trabalho.



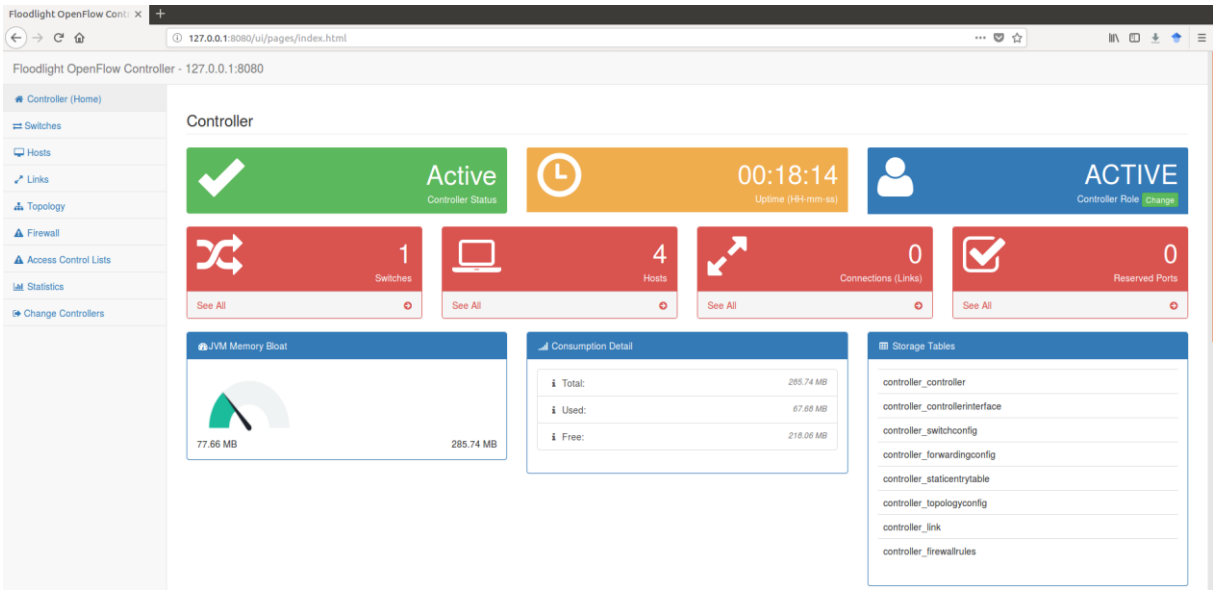
Nesse contexto, é importante destacar que, além da associação com as interfaces do *switch* S1, nenhuma configuração adicional foi necessária nos *hosts* H1, H2, H3 e H4 da rede.

4 RESULTADOS E DISCUSSÃO

Após a configuração dos elementos que compõem o cenário SDN OpenFlow de referência, o ambiente experimental está pronto para utilização. Ambos os *hosts* da rede estão acessíveis entre si (via comando *"ping"*, por exemplo) e é possível obter uma série de informações da SDN por meio da interface *web* fornecida pelo Floodlight (acessível via *url* http://127.0.0.1:8080/ui/pages/index.html no Linux Ubuntu) ou por meio de suas *northbound* REST APIs[7].

Como exemplo, a Figura 7 ilustra algumas informações obtidas sobre o ambiente SDN experimental por meio da interface *web* disponibilizada pelo Floodlight. De modo complementar, a Figura 8 ilustra um dos possíveis comandos para obtenção de informações sobre os dispositivos da SDN por meio das *northbound* REST APIs disponibilizadas pelo Floodlight.

Figura 7. Exemplo de informações obtidas sobre o ambiente SDN OpenFlow experimental por meio da interface *web* disponibilizada pelo Floodlight.



[7]Floodlight REST API em https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343539/Floodlight+REST+API

Figura 8. Exemplo de comando para obtenção de informações sobre os dispositivos do ambiente SDN OpenFlow experimental por meio das *northbound* REST APIs disponibilizadas pelo Floodlight.

```
curl http://127.0.0.1:8080/wm/device/ | python -m json.tool
```

Nesse cenário, também é possível utilizar as *northbound* REST APIs disponibilizadas pelo Floodlight para criar, listar e/ou remover regras de fluxos no ambiente SDN experimental. Como exemplo, a Figura 9 ilustra exemplos de comandos para a criação, listagem e remoção de regras de fluxos junto ao *switch* S1 por meio das *northbound* REST APIs disponibilizadas pelo Floodlight.

Figura 9. Exemplos de comandos para a criação, listagem e remoção de regras de fluxo junto ao *switch* S1 do ambiente SDN OpenFlow experimental por meio das *northbound* REST APIs disponibilizadas pelo Floodlight.

```
# Exemplo de comando para a criação da regra de fluxo "sample" no switch S1
curl    -X    POST    -d    '{"switch":"00:00:7a:95:85:9f:be:4f","name":"sample",
"active":"true","eth_type":"0x0800","ipv4_src":"172.17.0.1","ipv4_dst":"172.17.0.2
","actions":""}' http://127.0.0.1:8080/wm/staticentrypusher/json

# Exemplo de comando para a listagem de regras de fluxos no switch S1
curl   http://127.0.0.1:8080/wm/staticentrypusher/list/00:00:7a:95:85:9f:be:4f/json
| python -m json.tool

# Exemplo de comando para a remoção da regra de fluxo "sample" no switch S1
curl            -X            DELETE            -d            '{"name":"sample"}'
http://127.0.0.1:8080/wm/staticentrypusher/json | python -m json.tool
```

Nesse exemplo, o *switch* S1 é identificado pelo DPID *"00:00:7a:95:85:9f:be:4f"* (o qual pode ser obtido por meio do comando ilustrado na Figura 8, por exemplo) e a regra de fluxo *"sample"* é criada para realizar um *drop* dos pacotes transmitidos do *host* H1 para o *host* H2 (o *drop* é realizado automaticamente quando nenhuma ação é especificada durante a criação da regra de fluxo).

De modo complementar, também é possível obter informações e realizar ações específicas quanto ao *switch* S1 por meio dos comandos *"ovs*"* do Open vSwicth. Como exemplo, a Figura 10 ilustra exemplos de comandos para a obtenção de informações quanto ao *switch* S1 no âmbito do Open vSwitch, bem como para a criação e remoção de uma regra de fluxo similar à ilustrada na Figura 9.

Figura 10. Exemplos de comandos para a obtenção de informações quanto ao *switch* S1 no âmbito do Open vSwitch, bem como para a criação e remoção de uma regra de fluxo de exemplo.

```
# Exemplo de comandos para a obtenção de informações do switch S1/Open vSwitch
sudo ovs-vsctl show
sudo ovs-vsctl list-ports s1
sudo ovs-vsctl --version
sudo ovs-dpctl show
sudo ovs-ofctl show s1
sudo ovs-ofctl dump-flows s1

# Exemplo de comando para a criação de uma regra de fluxo no switch S1
sudo  ovs-ofctl  add-flow  s1  dl_type=0x0800,nw_src=172.17.0.3,nw_dst=172.17.0.4,
actions=

# Exemplo de comando para a remoção de uma regra de fluxo no switch S1
sudo ovs-ofctl del-flows s1 dl_type=0x0800,nw_src=172.17.0.3,nw_dst=172.17.0.4
```

Nesse exemplo, a regra de fluxo criada no *switch* S1 é responsável por realizar um *drop* dos pacotes transmitidos do *host* H3 para o *host* H4 (o *drop* é realizado automaticamente quando nenhuma ação é especificada durante a criação da regra de fluxo). Observe que essa mesma regra de fluxo é removida por meio do comando de remoção de regras de fluxo também ilustrado na Figura 10.

De modo complementar aos exemplos descritos nesta Seção, é importante destacar que as páginas oficiais do Floodlight e do Open vSwitch possuem excelentes documentações de referência,

com uma série de outros exemplos e comandos de referência de extrema valia no âmbito deste trabalho e das SDNs, de modo geral.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho discorreu sobre a prototipação de SDNs OpenFlow por meio de soluções de *software* amplamente utilizadas no âmbito das redes de computadores, as quais foram exploradas e discutidas a partir de um cenário SDN OpenFlow de referência configurado sobre o sistema operacional Linux Ubuntu. Em linhas gerais, essa discussão possibilitou ao leitor identificar algumas soluções e abordagens que podem ser empregadas para a prototipação e experimentação de SDNs OpenFlow, as quais podem ser utilizadas, por exemplo, em atividades de ensino e/ou de pesquisa relacionadas ao tema, bem como em novas contribuições no âmbito das redes de computadores, de modo geral.

Enquanto parte dos trabalhos futuros, objetiva-se a prototipação e experimentação de SDNs OpenFlow em outros cenários, tais como envolvendo questões no âmbito da segurança da informação e de comunicações multimídia, dentre outros.

REFERÊNCIAS

BI, Y. et al. Mobility Support for Fog Computing: An SDN Approach. **IEEE Communications Magazine**, v. 56, n. 5, p. 53-59, 2018.

DE OLIVEIRA, R. L. S. et al. Using Mininet for Emulation and Prototyping Software-Defined Networks. In: IEEE COLOMBIAN CONFERENCE ON COMMUNICATIONS AND COMPUTING (COLCOM), 2014, Bogota. **Proceedings...** Bogota: IEEE, 2014, p. 1-6.

DIORIO, R. F.; TIMÓTEO, V. S. Per-Flow Routing with QoS Support to Enhance Multimedia Delivery in OpenFlow SDN. In: PROCEEDINGS OF THE 22ND BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 2016, Teresina. **Proceedings...** Teresina: ACM, 2016. p. 167-174.

FONTES, R. et al. Mininet-WiFi: Emulating Software-Defined Wireless Networks. In: 11TH INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT (CNSM), 2015, Barcelona. **Proceedings...** Barcelona: IEEE, 2015, p. 384-389.

GUPTA, M.; SOMMERS, J.; BARFORD, P. Fast, Accurate Simulation for SDN Prototyping. In: PROCEEDINGS OF THE SECOND ACM SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, 2013, Hong Kong. **Proceedings...** Hong Kong: ACM, 2013, p. 31-36.

KREUTZ, D. et al. Software-Defined Networking: A Comprehensive Survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14-76, 2015.

MCKEOWN, N. et al. OpenFlow: Enabling Innovation in Campus Networks. **ACM SIGCOMM Computer Communication Review**, v. 38, n. 2, p. 69-74, 2008.

NUNES, B. A. A. et al. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. **IEEE Communications Surveys & Tutorials**, v. 16, n. 3, p. 1617-1634, 2014.

SHIRALI-SHAHREZA, S.; GANJALI, Y. Protecting Home User Devices with a SDN-based Firewall. **IEEE Transactions on Consumer Electronics**, 2018.

XIA, W. et al. A Survey on Software-Defined Networking. **IEEE Communications Surveys & Tutorials**, v. 17, n. 1, p. 27-51, 2015.

YAP, K. K. et al. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In: PROCEEDINGS OF THE CONFERENCE OF THE ACM SPECIAL INTEREST GROUP ON DATA COMMUNICATION, 2017, Los Angeles. **Proceedings...** Los Angeles: ACM, 2017, p. 432-445.