Implantação de um *Patch Panel* Virtual Utilizando Redes Definidas por *Software*

Whasley S. Cardoso
Instituto Federal da Paraíba - Campus Campina Grande
whasleycardoso@gmail.com

Fagner J. A. Silva
Instituto Federal da Paraíba - Campus Campina Grande
fagnerfjas@gmail.com

Uênio V. Rocha
Instituto Federal da Paraíba - Campus Campina Grande
uenio.v.rocha@ieee.org

Marcelo P. Sousa
Instituto Federal da Paraíba - Campus Campina Grande
marcelo.portela@ieee.org

ABSTRACT

Software Defined Networks enable significant gain in network management through separation of control and forwarding plans. In this work, the authors propose the implementation of a Virtual Patch Panel in order to restrict traffic to certain devices in an SDN approach, denoting the management flexibility and security optimization. GNS3 is used as a network simulation tool and OpenDayLight is used as the controller. The installation of rules in the controller is accomplished by different tools: OpenFlow Manager, Postman and cURL.

KEYWORDS

SDN, OpenFlow, OpenDayLight, Virtual Patch Panel.

1 INTRODUÇÃO

A Internet constitui serviços de fundamental importância nos dias atuais. As redes IP tradicionais são constituídas por diversos dispositivos, geralmente compostos por *software* fechado de determinado fabricante executado em *hardware* proprietário. Com isso, o desenvolvimento de novas tecnologias para este cenário é mais complexo. Atualmente, a configuração de regras na rede é realizada de maneira individual para cada dispositivo, utilizando os comandos de gerenciamento específicos de cada fabricante. Portanto, a capacidade de inovação e desenvolvimento de novas tecnologias para as redes atuais é minimizada, tornando-as, segundo [5], "engessada". Por outro lado, há o paradigma de SDN (Redes Definidas por *Software*) possibilitando uma abstração e centralização das operações e gerenciamento da rede. Este artigo apresenta uma integração de ferramentas possibilitando o gerenciamento e manipulação da tabela de fluxos de um *OpenVSwitch*.

2 SDN: VISÃO GERAL E TECNOLOGIAS

Nesta seção é apresentada uma visão geral dos principais conceitos acerca do paradigma de SDN e descreve as tecnologias *Southbound* e *Northbound* utilizadas neste trabalho.

In: XIV Workshop de Trabalhos de Iniciação Científica (WTIC 2017), Gramado, Brasil. Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: Workshops e Pôsteres. Porto Alegre: Sociedade Brasileira de Computação, 2017.

ISBN 978-85-7669-380-2.

2.1 Redes Definidas por *Software*

As Redes Definidas por *Software* são caracterizadas por meio da separação dos planos de controle e de encaminhamento de dados. O plano de controle é responsável pelas tomadas de decisão construindo e atualizando a tabela de encaminhamento, enquanto o plano de encaminhamento dados tem a função de realizar a comutação de pacotes. O plano de controle é desacoplado do plano de dados e implantado na forma do **controlador**, que é um sistema (*software*) que possui uma visão global da rede, podendo estar disposto local ou remotamente [1].

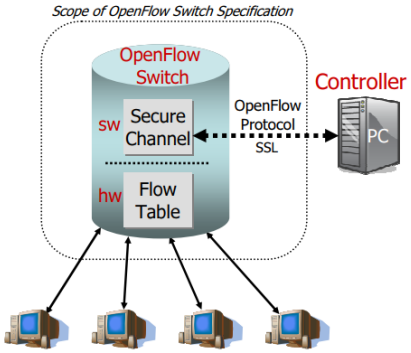

Figura 1: Arquitetura de Rede Definida por Software.

A Figura 1 ilustra uma arquitetura de rede utilizando um *Switch OpenFlow*, viabilizada a partir da definição do protocolo *OpenFlow* [7], que é melhor explorado na Seção 2.2. De maneira geral, essa definição pode ser utilizada para representar a arquitetura SDN, onde a separação dos planos é bem definida. A partir desta separação de planos, o dispositivo habilitado para utilizar o protocolo *OpenFlow* é denominado *Switch OpenFlow* e seu modo de operação é determinado pelo administrador da rede inserindo regras por meio de ambientes de programação de alto nível [4].

2.2 *Southbound: OpenFlow*

O *OpenFlow* é um protocolo aberto apresentado em [7] como uma solução à problemática da inviabilidade de realizar pesquisas utilizando os equipamentos proprietários. Com desenvolvimento inicialmente na Universidade de *Stanford* e da Universidade da Califórnia e posteriormente mantido pela ONF (*Open Networking Foundation*),

o *OpenFlow* é o padrão comumente utilizado no contexto de Redes Definidas por *Software*.

O *OpenFlow* faz parte da camada *Southboud*, tendo a função de realizar a comunicação entre o controlador e o *switch* . Uma regra enviada do controlador para ser instalada no *switch OpenFlow* tem um formato bem definido, abrangendo diversas possibilidades de controle. A Figura 2 ilustra um exemplo do formato de uma tabela de fluxos.

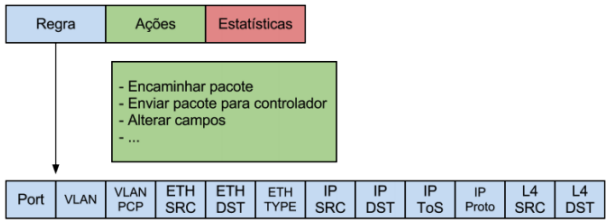

Figura 2: Exemplo de uma tabela de fluxos *OpenFlow* .

Com base na Figura 2 é possível inserir uma regra, por exemplo, para descartar um pacote com destino para determinado endereço IP. Tal configuração pode ser realizada inserindo no campo regra ou *Match*, o endereço IP destino ao qual um administrador de rede pretende impedir o tráfego, enquanto no campo Ações é inserido um parâmetro como *Drop*. Com isso, o tráfego com destino ao endereço IP informado é descartado. Portanto, o *OpenFlow* fornece uma diversidade considerável de possibilidades de controle. Além disso, uma vez que o controlador SDN possui uma visão global da topologia, é possível gerenciar vários *Switches OpenFlow* de maneira centralizada.

A inserção de um fluxo no *Switch OpenFlow* é realizada por meio do controlador, utilizando a camada *Southbound* representada, neste caso, pelo padrão *OpenFlow*. Porém, do ponto de vista do administrador da rede, esta configuração é de mais baixo nível tornando a tarefa mais complexa. No entanto, a arquitetura SDN apresenta também, a camada *Northbound*. O padrão comumente utilizado é o REST (*REpresentational State Transfer*) [2]. O *REST-API* fornece um ambiente de mais alto nível para a comunicação com o controlador SDN.

2.3 *Northbound: REST-API*

Aplicações *Northbound* são responsáveis pela comunicação entre aplicações de alto nível e o controlador SDN, por meio de um ambiente de programação. Dentre as aplicações que realizam esta função, *REST-API* destaca-se no contexto de Redes Definidas por *Software*. *REST* utiliza o protocolo *HTTP* em sua comunicação e trabalha essencialmente com as requisições Web: *GET, POST, PUT, DELETE*, entre outros[2, 6]. O formato das requisições é bem definido utilizando, também, *JSON* e *XML*. As Figuras 3 e 4 ilustram trechos de códigos representados nesses formatos.

Neste trabalho foram exploradas três ferramentas para inserção de fluxos no controlador, ambas utilizando *REST*. As aplicações utilizadas são: *Postman*[1], OFM (*OpenFlow Manager*) [2] e *cURL*[3].

[1]https://www.getpostman.com/
[2]https://github.com/CiscoDevNet/OpenDaylight-Openflow-App
[3]https://curl.haxx.se/

```
<router>
 <fabricante>Cisco</fabricante>
 <modelo>c7000</modelo>
 <id>15</id>
<router>
```

Figura 3: Exemplo de código no formato XML.

```
{
    "fabricante": "Cisco",
    "modelo": "c700",
    "id": "15"
}
```

Figura 4: Exemplo de código no formato JSON.

2.4 Controlador SDN

Um dos principais componentes de uma Rede Definida por *Software* é o controlador, também chamado de sistema operacional de rede. O controlador é o que define a natureza do paradigma SDN. É o componente responsável por concentrar a comunicação com todos os elementos programáveis da rede, oferecendo uma visão unificada da topologia.

Existem diversos controladores desenvolvidos em diferentes linguagens que dão suporte ao paradigma SDN. Neste trabalho é utilizado o ODL (OpenDayLight), um projeto *Open Source* que provê uma plataforma para controle em SDN, criado em 2013 e mantido pela *Linux Foundation*. ODL oferece uma plataforma aberta e modular, para customizar e automatizar redes de pequeno, médio e grande porte.

Desenvolvido na linguagem **java**, este controlador conta com uma grande comunidade de desenvolvedores. O ODL possibilita o controle e configuração dos dispositivos da rede por meio do protocolo *OpenFlow* e possui módulos, que podem ser habilitados. Entre suas funções, há uma interface gráfica Web (GUI), em que é possível visualizar a topologia, monitorar os fluxos ativos nos dispositivos e configurá-los. A interação entre aplicações de alto nível e o controlador é realizada utilizando *REST-API* [3].

3 CENÁRIO

Para a implantação de um *Patch Panel* virtual foi utilizada a topologia ilustrada na Figura 5, preparada no GNS3[4]. O cenário está disposto de maneira que um *Switch OpenFlow* (*OpenVSwitch-1*) está interligado com quatro roteadores Cisco. O objetivo é programar o *Switch OpenFlow* de modo que este funcione como um *Patch Panel*, direcionando o tráfego para portas específicas, de maneira individual. O controle da rede está disposto em uma máquina virtual (Controlador) com o *OpenDayLight* e o *OpenFlow Manager* instalados. Uma máquina de gerenciamento foi inserida para acessar os serviços e enviar as requisições ao controlador.

Com base neste cenário, quatro regras são inseridas no controlador, de modo que R1 e R2 devem trafegar pacotes entre si, apenas.

[4]https://www.gns3.com/

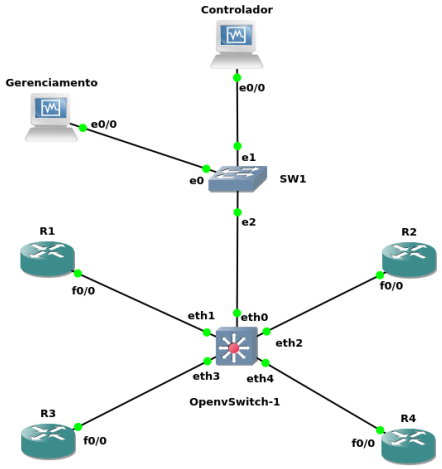

Figura 5: Topologia para o *Patch Panel* Virtual no GNS3.

O mesmo deve ocorrer entre R3 e R4. A Tabela 1 apresenta o relacionamento entre os roteadores e as interfaces correspondentes do *OpenVSwitch-1*.

Tabela 1: Conexão dos dispositivos na topologia.

Roteador	Interface do *OpenVSwitch*
R1	2
R2	3
R3	4
R4	5

4 RESULTADOS

4.1 Inserção de fluxos utilizando o *OpenFlow Manager*

O *OpenFlow Manager* fornece uma interface para gerenciamento da tabela de fluxos de um *Switch OpenFlow* . Para permitir a comunicação entre R1 e R2 é necessário inserir duas regras no controlador: (*i*) Habilitar a comunicação no sentido R1 - R2; (*ii*) Habilitar a comunicação de retorno, no sentido R2 - R1. O mesmo procedimento deve ser realizado com a conexão entre R3 e R4, somando para este cenário, quatro regras instaladas no *OpenVSwitch*.

Com base na Tabela 1, a regra para habilitar a comunicação no sentido R3 - R4 consiste na seguinte lógica: pacotes que chegam à interface 4, devem ser encaminhados pela interface 5. O formato JSON desta regra é ilustrado na Figura 6.

A Figura 7 mostra o gerenciamento de fluxos do *OpenFlow Manager* com as regras instaladas.

Durante a instalação do fluxo é possível analisar a troca de mensagens, utilizando o *wireshark*, capturando os pacotes trafegados entre o *OpenFlow Manager* e o ODL. A Figura 8 ilustra uma captura contendo a requisição *PUT* para a inserção do fluxo no *OpenVSwitch*, em que a origem http://192.168.0.55:9000 representa o serviço *OpenFlow Manager*.

Uma vez instaladas as regras no *OpenVSwitch*, a implantação do *Patch Panel* virtual está concluída. Neste momento, só existe

```
{
    "flow": [
        {
            "table_id": "0",
            "id": "4-5",
            "priority": "1000",
            "match": {
                "in-port": "openflow:205314561770572:4"
            },
            "instructions": {
                "instruction": [
                    {
                        "order": 0,
                        "apply-actions": {
                            "action": [
                                {
                                    "order": 0,
                                    "output-action": {
                                        "output-node-connector": "5",
                                        "max-length": "65535"
                                    }
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

Figura 6: Configuração da regra no formato JSON.

Flow name	ID	Table ID	Device	Device type
[id:5-4, table:0]	5-4	0	openflow:205314561770572	Open vSwitch
[id:4-5, table:0]	4-5	0	openflow:205314561770572	Open vSwitch
[id:2-3, table:0]	2-3	0	openflow:205314561770572	Open vSwitch
[id:3-2, table:0]	3-2	0	openflow:205314561770572	Open vSwitch

Figura 7: Regras instaladas utilizando o *OpenFlow Manager*.

```
▼Hypertext Transfer Protocol
 ▶PUT /restconf/config/opendaylight-inventory:nodes
  Host: 192.168.0.55:8181\r\n
  Connection: keep-alive\r\n
 ▶Content-Length: 333\r\n
  Accept: application/json, text/plain, */*\r\n
  Origin: http://192.168.0.55:9000\r\n
```

Figura 8: Pacote capturado durante a instalação do fluxo.

```
/ # ovs-ofctl -O OpenFlow13 dump-flows br0
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=2386.964s, table=0, n_packets=155, n_bytes=16289, priority=1000,in_port=2 actions=output:3
 cookie=0x0, duration=2386.964s, table=0, n_packets=189, n_bytes=20066, priority=1000,in_port=4 actions=output:5
 cookie=0x0, duration=2386.964s, table=0, n_packets=155, n_bytes=16360, priority=1000,in_port=3 actions=output:2
 cookie=0x0, duration=1399.772s, table=0, n_packets=111, n_bytes=11336, priority=1000,in_port=5 actions=output:4
 cookie=0x2b0000000000001, duration=2386.964s, table=0, n_packets=0, n_bytes=0, priority=100,dl_type=0x88cc actl
 cookie=0x0, duration=2386.964s, table=0, n_packets=183, n_bytes=27896, priority=2 actions=CONTROLLER:65535
 cookie=0x2b0000000000001, duration=2386.963s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
/ #
```

Figura 9: Tabela de fluxos do *OpenVSwitch* com as regras instaladas.

comunicação entre R1 com R2 e R3 com R4. A Figura 10 confirma a conectividade entre R1 e R2. No entanto, a partir da Figura 11 nota-se que não existe conectividade entre R1 e R4, mesmo estando na mesa rede.

```
R1#ping 172.16.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/44/48 ms
R1#
```

Figura 10: Conectividade entre R1 e R2.

```
R1#ping 172.16.0.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.4, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R1#
```

Figura 11: Conectividade entre R1 e R4.

4.2 Inserção de fluxos utilizando o *Postman*

Figura 12: Inserindo uma regra utilizando o *Postman*.

A ferramenta *Postman* fornece um ambiente de fácil gerenciamento para manipulação de requisições Web. A Figura 12 apresenta um exemplo de inserção de uma regra no controlador utilizando a requisição *PUT*.

4.3 Inserção de fluxos utilizando *cURL*

cURL é uma ferramenta para manipulação de URLs e com ela é possível obter ou enviar arquivos utilizando os protocolos suportados, dentre esses, o HTTP. A partir de uma linha de comando e um arquivo com extensão **json** é possível inserir uma regra no controlador a ser instalada no *OpenVSwitch* .

Para a inserção de um fluxo utilizando o *cURL*, a seguinte linha de comando foi utilizada:

curl -u admin:admin -H 'Content-Type: application/json' -X PUT -d @fluxo.json <caminho>[5]

Na linha de comando especifica-se os parâmetros de autenticação, formato do arquivo que será enviado, neste caso *application/json* e um arquivo no formato especificado contendo a regra. Por fim,

[5]http://<controllerIP>:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1

```
Hypertext Transfer Protocol
  PUT /restconf/config/opendaylight-inventory:nodes/node/openflow:2053145617
    [Expert Info (Chat/Sequence): PUT /restconf/config/opendaylight-inventory
    Request Method: PUT
    Request URI: /restconf/config/opendaylight-inventory:nodes/node/openflow:
    Request Version: HTTP/1.1
  Host: 192.168.0.55:8181\r\n
  Authorization: Basic YWRtaW46YWRtaW4=\r\n
  User-Agent: curl/7.47.0\r\n
```

Figura 13: Regra inserida utilizando *cURL*.

é acrescentado o caminho no qual as regras são armazenadas no controlador. Este caminho pode variar de acordo com os parâmetros da topologia e da configuração da regra.

5 CONCLUSÃO E TRABALHOS FUTUROS

A partir dos resultados obtidos é possível destacar a diversidade de possibilidades de controle. Neste trabalho, uma configuração de *Patch Panel* virtual foi implantada no contexto de Redes Definidas por *Software*, restringindo o tráfego apenas entre dispositivos específicos. Em uma configuração convencional de VLANs, por exemplo, um pacote com destino a um *host* A, pode chegar a um *host* B sendo descartado por este. Portanto, além de um ganho na flexibilidade de gerenciamento da rede, a configuração proposta neste trabalho apresenta também um ganho de segurança. Como continuação deste trabalho, os autores pretendem explorar parâmetros de Engenharia de Tráfego no contexto de Redes Definidas por *Software*, utilizando dispositivos reais.

REFERÊNCIAS

[1] Whasley S. Cardoso e Marcelo P. Sousa Edlane O. G. Alves, Marcela T. G. Santos. 2016. Uso de Redes Definidas por Softwarepara Controle de Roteamento. *Instituto de Estudos Avançados em Comunicações (Iecom)* (Outubro 2016).

[2] Roy Thomas Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures. *University of California, Irvine* (April 2000).

[3] Linux Foundation. 2017. *About The OpenDaylight Foundation.* https://www.opendaylight.org/about.

[4] Dorgival Guedes Henrique Rodrigues e Rogerio V. Nunes Luiz Filipe M. Vieira, Marcos M. Vieira. 2012. Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento das pesquisas em Redes de Computadores. *Departamento de Ciência da Computação – Universidade Federal de Minas Gerais* (March 2012).

[5] Kreutz D. Veríssimo P. E. Rothenberg C. E. Azodolmolky S. Ramos, F. M. V. and S. Uhlig. 2015. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (Junho 2015), 14–76.

[6] Myung-Ki Shin, Ki-Hyuk Nam, and Hyoung-Jun Kim. 2012. Software-defined networking (SDN): A reference architecture and open APIs. In *ICT Convergence (ICTC), 2012 International Conference on.* IEEE, 360–361.

[7] Nick McKeown Guru Parulkar Larry Peterson Jennifer Rexford Scott Shenker Tom Anderson, Hari Balakrishnan and Jonathan Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *Stanford University* (March 2008).