

	Universidade Federal de Pernambuco Centro de Informática Redes de Computadores	
	Relatório Final do Projeto 1	
	Aluno: Arnaldo Rafael Morais Andrade	Data: 28/11/2018

1) Introdução

Este projeto tem como objetivo implementar um conjunto de encaminhamento e políticas de alocação de banda dentro de uma rede, utilizando um controlador remoto. A implementação atuará sob uma rede virtual instanciada pelo Mininet. Para o objetivo em questão, foi utilizado o controlador POX, por motivos de melhores documentações encontradas. Ainda por cima, este controlador já foi visto em aulas da monitoria.

2) Conceitos

2.1) SDN

Software-Defined Network (ou Redes definidas por software) É uma abordagem que visa desacoplar as funções do hardware que as implementa e executá-las em Software. O controle da rede é programável diretamente pelos administradores, permitindo configurar, gerenciar, proteger e otimizar os recursos de rede muito rapidamente através de dinâmicas e programas SDN automatizados.

É permitido que os administradores ajustem dinamicamente o fluxo de tráfego em toda a rede para atender as demandas de mudança. Além disso, toda inteligência da rede é logicamente centralizada nos controladores baseados em software que possuem um visão global da rede.

2.2) OpenFlow 1.3

Este padrão descreve um protocolo aberto que permite que aplicações em software possam coordenar as tabelas de fluxo de diferentes dispositivos. O OpenFlow explora conjuntos comuns de funções que são utilizadas em switches e roteadores.

Os seus 3 principais elementos: Tabela de Fluxo em que suas entradas possuem 3 partes. 1. Regras, 2.Ações, 3.Estatísticas. Que respectivamente, identificam fluxos através de

padrões, guardam informações do que fazer com cada pacote de fluxo e conta os bytes e pacotes do fluxo. No campo de regras pode ter campos como: Port, VLAN, ETH, IP, L4, MAC...

Canal Seguro, responsável pela comunicação entre o switch e o controlador. A tabela de fluxo é controlada por um controlador remoto, via este canal.

E por fim o Protocolo OpenFlow que é uma proposta para padronizar a comunicação entre um elemento externo (comumente baseado em software), denominado controlador, e os dispositivos de redes (switch OpenFlow) na arquitetura SDN.

A especificação 1.3 do OpenFlow adicionou significativas funcionalidades ao protocolo. Quase todas foram implementadas, com exceção apenas de conexões auxiliares utilizando o protocolo UDP [1].

Vale destacar alguns componentes exclusivos desta versão: **Ofdatapath**: a implementação do switch; **Ofprotocol**: canal seguro para conectar o switch ao controlador; **Oflib**: uma biblioteca para converter de/para o fio de formato 1.3; **Dpctl**: uma ferramenta para configurar o switch a partir do console [2].

2.3) POX

POX é um controlador Openflow, bastante leve, escrito completamente em Python, para desenvolvedores montarem seu próprio controlador.

Ele provê um framework para comunicação com switches SDN usando o protocolo OpenFlow ou OVSDB.

O POX também pode ser usado como um controlador básico SDN usando apenas componentes integrados à ferramenta.

3) Roteiro

3.1) Metodologia

Para emular as imagens, foi utilizado o Virtual Box. A imagem utilizada para o projeto foi uma 64-bit Ubuntu 14.04, que já continha todos os softwares necessários [3].

Já dentro da máquina virtual, fez-se o clone do repositório git, o qual estava o projeto (<https://github.com/arma29/Redes.git>). Com o terminal emulado aberto, navegou-se até o

diretório Redes/Projeto e o executou da seguinte forma:

```
$ sudo -E python mytopo.py
```

Em outro terminal, no mesmo diretório, primeiramente foi feita a cópia do script POX para o diretório original do controlador:

```
$ cp poxproj.py ~/pox/ext
```

Finalizada a cópia, bastou executar o script POX dentro da pasta.

```
$ cd ~/pox
```

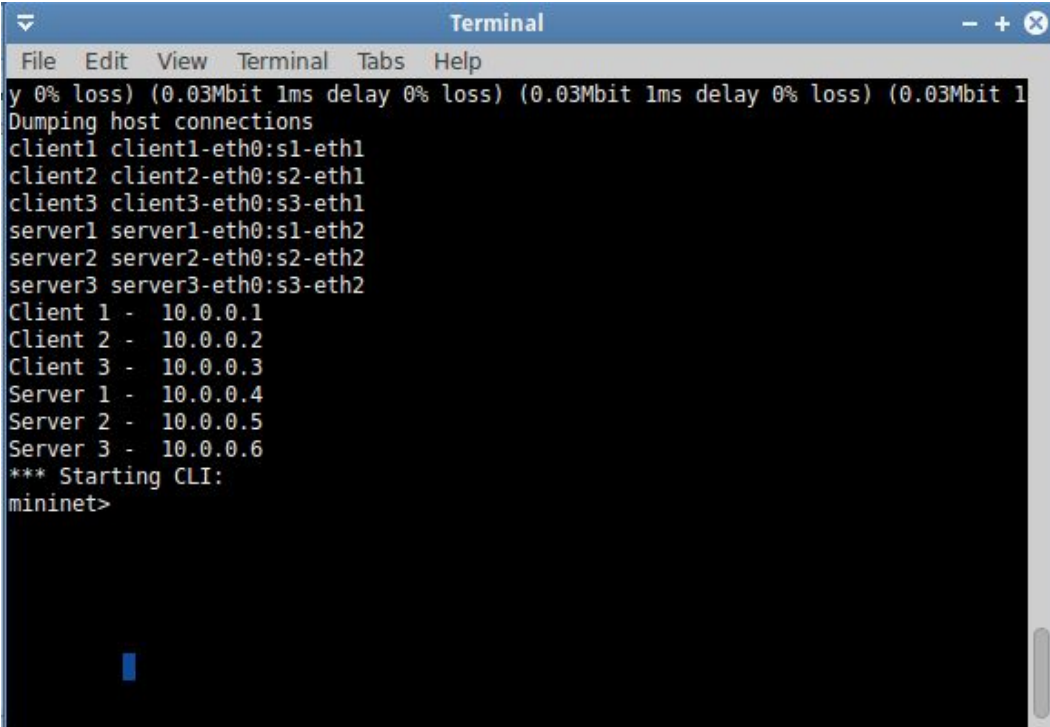
```
$ python pox.py --verbose poxproj.py log --no-default --file=/tmp/mylog.log
```

Com este roteiro, a rede instanciada já está pronta para alguns testes. Os Resultados são mostrados na seção abaixo.

4) Resultados Parciais

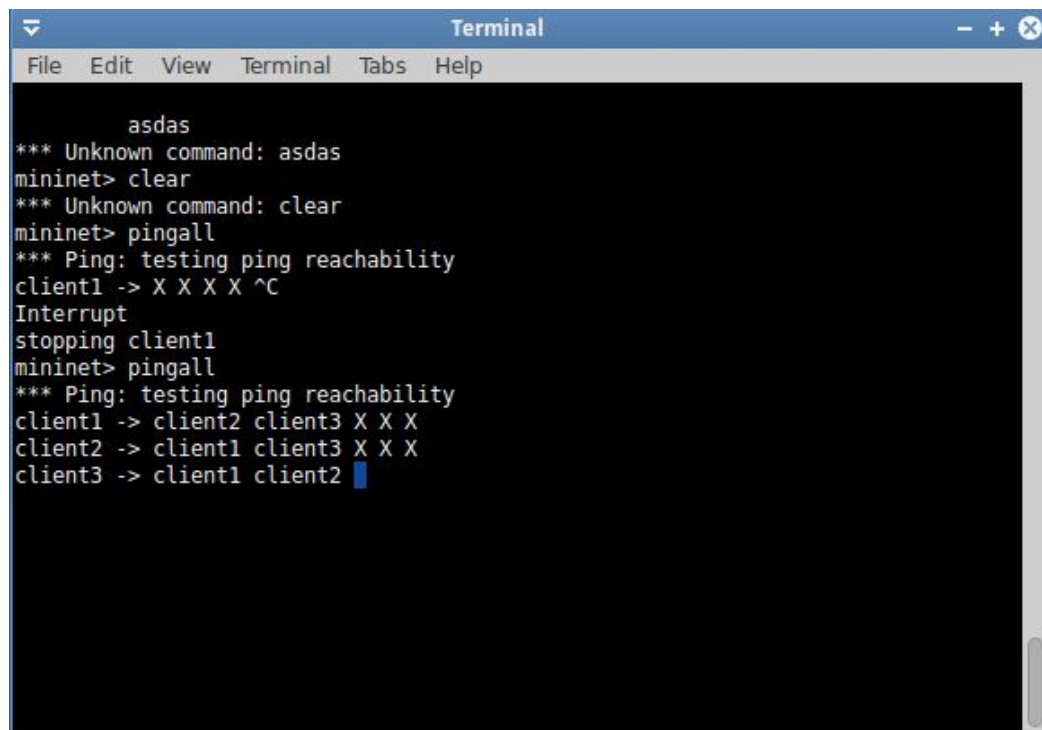
Os testes foram feitos em um Desktop com sistema operacional Ubuntu 16.04, Processador Intel® Core™ i7-3770 CPU @ 3.40GHz × 8 , Memória 7,7 GiB, Gráficos Intel® Ivybridge Desktop. Compilador utilizado GCC/G++ 5.4.0.

Os resultados são apresentados em formas de figura, no próprio terminal da máquina virtual. Comandos como *pingall* e *ping* par a par, foram utilizados.

A screenshot of a terminal window titled "Terminal" with standard window controls. The terminal displays the output of a network configuration script. It lists three clients and three servers with their respective interface names and IP addresses. The output ends with the Mininet CLI prompt.

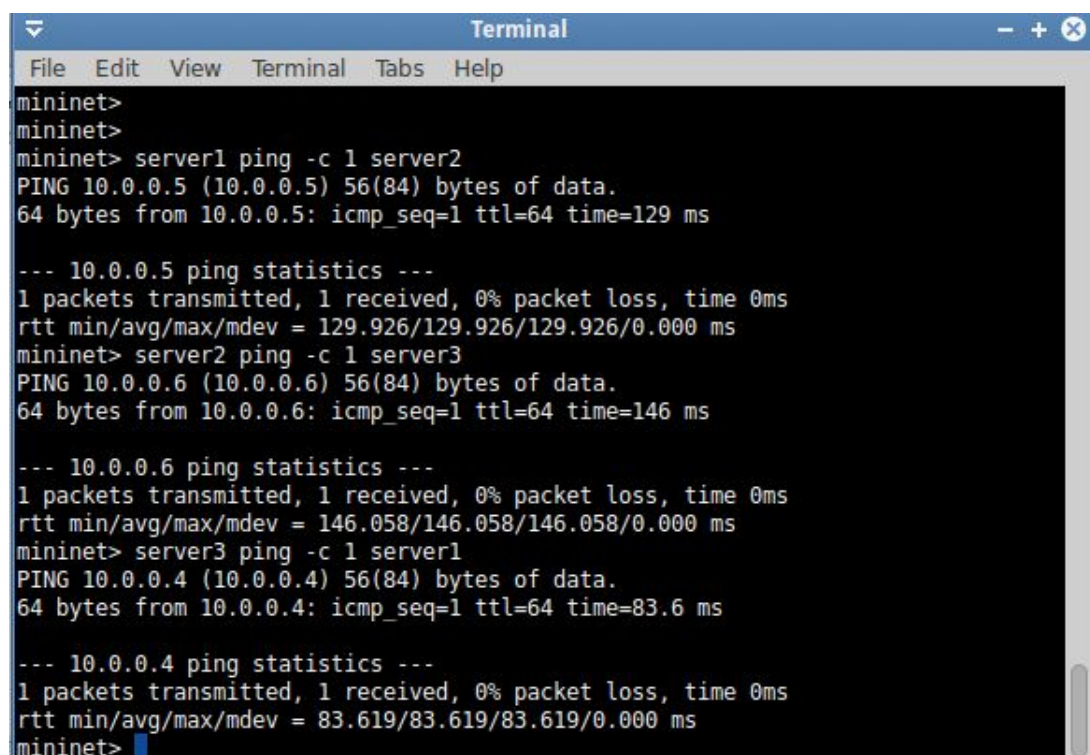
```
File Edit View Terminal Tabs Help
y 0% loss) (0.03Mbit 1ms delay 0% loss) (0.03Mbit 1ms delay 0% loss) (0.03Mbit 1
Dumping host connections
client1 client1-eth0:s1-eth1
client2 client2-eth0:s2-eth1
client3 client3-eth0:s3-eth1
server1 server1-eth0:s1-eth2
server2 server2-eth0:s2-eth2
server3 server3-eth0:s3-eth2
Client 1 - 10.0.0.1
Client 2 - 10.0.0.2
Client 3 - 10.0.0.3
Server 1 - 10.0.0.4
Server 2 - 10.0.0.5
Server 3 - 10.0.0.6
*** Starting CLI:
mininet>
```

Figura 1 - Rede Instanciada pelo Mininet



```
asdas
*** Unknown command: asdas
mininet> clear
*** Unknown command: clear
mininet> pingall
*** Ping: testing ping reachability
client1 -> X X X X ^C
Interrupt
stopping client1
mininet> pingall
*** Ping: testing ping reachability
client1 -> client2 client3 X X X
client2 -> client1 client3 X X X
client3 -> client1 client2
```

Figura 2 - Teste de conexão entre clientes (pingall com abortagem)



```
mininet>
mininet>
mininet> server1 ping -c 1 server2
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=129 ms

--- 10.0.0.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 129.926/129.926/129.926/0.000 ms
mininet> server2 ping -c 1 server3
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=146 ms

--- 10.0.0.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 146.058/146.058/146.058/0.000 ms
mininet> server3 ping -c 1 server1
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=83.6 ms

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 83.619/83.619/83.619/0.000 ms
mininet>
```

Figura 3 - Teste de conexão entre servidores no sentido descrito no projeto.

5) Conclusão

Da implementação parcial ao projeto final, não ocorreram avanços. O ONOs era bastante denso, se buildado pela vm[3]. Não foi executado com perfeição

Os resultados apresentados são insatisfatórios e mostram apenas a parte estática do projeto. A parte dinâmica não foi realizada.

6) Referências

- [1] <https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/fernandes2014openflow.pdf>
- [2] <http://cpqd.github.io/ofsoftswitch13/>
- [3] <http://sdnhub.org/tutorials/sdn-tutorial-vm/>