Universidade Federal da Bahia
MATE18 – Tópicos em Redes de Computadores III
Oficina de OpenFlow/SDN

Professor: Leobino Sampaio <leobino@ufba.br>
Instrutor: Italo Valcy <italovalcy@ufba.br>

Aula 02 – Controlador OpenFlow

Observação: ao longo das práticas aqui descritas o aluno será convidado a executar comandos em três ambientes diferentes:

- No Shell do Linux (aqui representado pela string "`mininet@oficina-openflow-ufba:~$`")
- No console do Mininet (aqui representado pela string "`mininet>`")
- No console do POX (aqui representado pela string "`POX>`")

Prática 01 – Configuração de fluxos OF via console de aplicação

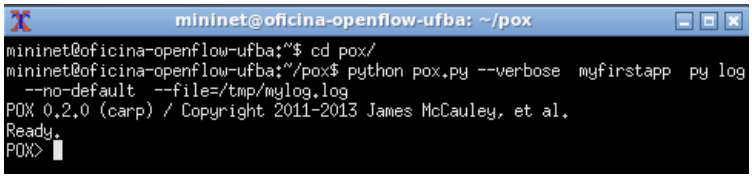
1. O primeiro passo é criar a aplicação "myfirstapp" e armazenar ela no arquivo ~/pox/ext/myfirstapp.py. Para isso, devemos executar o seguinte comando para fazer o download da aplicação no local correto:

```
cd ~/pox/ext/
wget http://homes.dcc.ufba.br/~italo/oficina-openflow-2017/myfirstapp.py
```

2. Abrir um terminal "xterm" e executar o seguinte comando para iniciar o controlador OpenFlow:

```
cd ~/pox
python pox.py --verbose myfirstapp py log --no-default --file=/tmp/mylog.log
```

Ao executar esse comando, a saída esperada é a seguinte:



Como utilizamos o módulo de logging do POX e salvando os dados em /tmp/mylog.log, podemos monitorar as mensagens do controlador através do seguinte comando (executado em outro terminal):

```
tail  -f /tmp/mylog.log
```

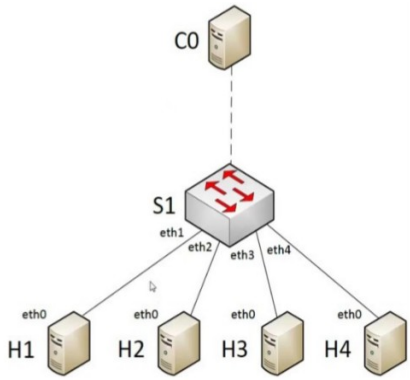
Outra coisa interessante é monitorar as mensagens OF via Wireshark (veja no roteiro da aula 01 para detalhes de como executá-lo).

3. Em seguida, vamos abrir um novo terminal "xterm" e iniciar a topologia via Mininet. Para isso, devemos executar o seguinte comando:

```
sudo mn --topo single,4 --mac --arp --controller remote
```

Ao executar esse comando, a saída esperada (esquerda) e a topologia criada (direita) são:

```
mininet@oficina-openflow-ufba: ~
mininet@oficina-openflow-ufba:~$ sudo mn --topo single,4 --mac --arp --controlle
r remote
*** 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
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```



4. É necessário agora configurar as regras OF a partir do console do controlador no switch em questão. Para isso, executamos os seguintes comandos no console do POX:

```
import pox.openflow.libopenflow_01 as of
from myfirstapp import myfirstapp

msg = of.ofp_flow_mod()
msg.match.in_port = 1
msg.actions.append(of.ofp_action_output(port = 3))
myfirstapp.switches[1].send(msg)

msg = of.ofp_flow_mod()
msg.match.in_port = 3
msg.actions.append(of.ofp_action_output(port = 1))
myfirstapp.switches[1].send(msg)
```

Após a execução com sucesso dos comandos, você deve visualizar o seguinte:

```
POX> import pox.openflow.libopenflow_01 as of
POX> from myfirstapp import myfirstapp
POX>
POX> msg = of.ofp_flow_mod()
POX> msg.match.in_port = 1
POX> msg.actions.append(of.ofp_action_output(port = 3))
POX> myfirstapp.switches[1].send(msg)
POX>
POX> msg = of.ofp_flow_mod()
POX> msg.match.in_port = 3
POX> msg.actions.append(of.ofp_action_output(port = 1))
POX> myfirstapp.switches[1].send(msg)
POX>
```

É interessante checar também na tabela de fluxos do Switch1 (s1) as regras recém criadas (via console do Mininet):

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=105.323s, table=0, n_packets=0, n_bytes=0, idle_age=105, i
n_port=3 actions=output:1
 cookie=0x0, duration=106.22s, table=0, n_packets=0, n_bytes=0, idle_age=106, in
_port=1 actions=output:3
mininet>
```

5. Por fim, podemos realizar um teste de conectividade entre o H1 e H3 ou demais nós. Apenas a comunicação entre H1 e H3 devem funcionar.

```
mininet> h1 ping -c3 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.973 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.337 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.298 ms

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.298/0.536/0.973/0.309 ms
mininet>
```

Prática 02 – Desenvolver lógica do controlador OpenFlow descarte de pacotes

Nessa prática o objetivo é criar uma aplicação no POX para fazer o descarte de todos os pacotes recebidos pelo switch nas portas OpenFlow. A figura abaixo demonstra a tabela de fluxos esperada:

	Match												Ação
Prioridade	Porta de entrada	MAC de origem	MAC de destino	Tipo Ethernet	VLAN ID	VLAN Priority	IP de origem	IP de destino	ToS	Protocolo	Porta de origem	Porta de destino	Ação
-	1	-	-	-	-	-	-	-	-	-	-	-	drop
-	2	-	-	-	-	-	-	-	-	-	-	-	drop
-	3	-	-	-	-	-	-	-	-	-	-	-	drop
-	4	-	-	-	-	-	-	-	-	-	-	-	drop

1. O primeiro passo é criar a aplicação “aula2ex2” e armazenar ela no arquivo ~/pox/ext/aula2ex2.py. Para isso, devemos executar o seguinte comando para fazer o download da aplicação no local correto:

```
cd ~/pox/ext/
wget http://homes.dcc.ufba.br/~italo/oficina-openflow-2017/aula2ex2.py
```

2. Abrir um terminal “xterm” e executar o seguinte comando para iniciar o controlador OpenFlow:

```
cd ~/pox
python pox.py --verbose  aula2ex2  py log  --no-default  --file=/tmp/mylog.log
```

3. Em seguida, vamos abrir um novo terminal “xterm” e iniciar a topologia via Mininet. Para isso, devemos executar o seguinte comando:

```
sudo mn --topo single,4 --mac --arp --controller remote
```

4. Para confirmar que a aplicação está funcionando, ou seja, que a aplicação cria regras de descarte (drop) para todas as requisições recebidas nas portas OF do switch, execute o seguinte comando no console do Mininet e observe que os hosts não terão conectividade:

```
pingall
```

A saída esperada é:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
mininet>
```

5. É interessante também observarmos a tabela de fluxos do switch1 (s1) para confirmar se está descartando todos os pacotes conforme objetivo desta tarefa. Para isso, execute o seguinte comando no console do Mininet:

```
sh ovs-ofctl dump-flows s1
```

A saída esperada é:

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=194.386s, table=0, n_packets=0, n_bytes=0, idle_age=194, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=174.357s, table=0, n_packets=0, n_bytes=0, idle_age=174, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=drop
 cookie=0x0, duration=124.186s, table=0, n_packets=0, n_bytes=0, idle_age=124, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=144.25s, table=0, n_packets=0, n_bytes=0, idle_age=144, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:01 actions=drop
 cookie=0x0, duration=154.279s, table=0, n_packets=0, n_bytes=0, idle_age=154, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:04 actions=drop
 cookie=0x0, duration=224.456s, table=0, n_packets=0, n_bytes=0, idle_age=224, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=184.34s, table=0, n_packets=0, n_bytes=0, idle_age=184, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04 actions=drop
 cookie=0x0, duration=134.261s, table=0, n_packets=0, n_bytes=0, idle_age=134, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02 actions=drop
 cookie=0x0, duration=214.425s, table=0, n_packets=0, n_bytes=0, idle_age=214, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04 actions=drop
 cookie=0x0, duration=164.323s, table=0, n_packets=0, n_bytes=0, idle_age=164, in_port=3,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=drop
 cookie=0x0, duration=234.491s, table=0, n_packets=0, n_bytes=0, idle_age=234, in_port=1,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=drop
 cookie=0x0, duration=204.441s, table=0, n_packets=0, n_bytes=0, idle_age=204, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=drop
mininet>
```

Prática 03 – Desenvolver lógica do controlador para encaminhar tráfego entre portas 1 e 3, descartando as demais

Nessa prática o objetivo é criar uma aplicação no POX para fazer o encaminhamento de pacotes entre as portas 1 e 3 e descartar todos os pacotes recebidos nas demais portas nas portas OF do switch. A figura abaixo demonstra a tabela de fluxos esperada:

Match													Ação
Prioridade	Porta de entrada	MAC de origem	MAC de destino	Tipo Ethernet	VLAN ID	VLAN Priority	IP de origem	IP de destino	ToS	Protocolo	Porta de origem	Porta de destino	Ação
-	1	-	-	-	-	-	-	-	-	-	-	-	Output:3
-	2	-	-	-	-	-	-	-	-	-	-	-	drop
-	3	-	-	-	-	-	-	-	-	-	-	-	Output:1
-	4	-	-	-	-	-	-	-	-	-	-	-	drop

1. O primeiro passo é criar a aplicação “aula2ex3” e armazenar ela no arquivo ~/pox/ext/aula2ex3.py. Para isso, devemos executar o seguinte comando para fazer o download da aplicação no local correto:

```
cd ~/pox/ext/
wget http://homes.dcc.ufba.br/~italo/oficina-openflow-2017/aula2ex3.py
```

Observe as diferenças entre a aplicação aula2ex2 e aula2ex3. Note como a porta de entrada no switch foi utilizada para a condicional de execução e também o valor utilizado na action.

2. Abrir um terminal “xterm” e executar o seguinte comando para iniciar o controlador OpenFlow:

```
cd ~/pox
python pox.py --verbose  aula2ex3  py log  --no-default  --file=/tmp/mylog.log
```

3. Em seguida, vamos abrir um novo terminal “xterm” e iniciar a topologia via Mininet. Para isso, devemos executar o seguinte comando:

```
sudo mn --topo single,4 --mac --arp --controller remote
```

4. Vamos testar a conectividade entre os H1 e H3. Para isso, execute o seguinte comando no console do Mininet:

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.724 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.252 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.166 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.464 ms
^C
--- 10.0.0.3 ping statistics ---
6 packets transmitted, 4 received, 33% packet loss, time 5004ms
rtt min/avg/max/mdev = 0.166/0.401/0.724/0.216 ms
mininet>
```

Observe a perda de dois (02) pacotes no teste ICMP. Por que isso ocorre? Se você repetir o teste, perde também?

5. Para confirmar que a aplicação está funcionando, ou seja, que a aplicação cria regras de descarte (drop) para todas as requisições recebidas nas portas OF do switch, execute o seguinte comando no console do Mininet e observe que os hosts não terão conectividade:

```
pingall
```

A saída esperada é:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X
h2 -> X X X
h3 -> h1 X X
h4 -> X X X
*** Results: 83% dropped (2/12 received)
```

Observe que apenas o nó h1 consegue efetuar PING para h3 e vice-versa.

6. É interessante também observarmos a tabela de fluxos do switch1 (s1) para confirmar se está descartando todos os pacotes conforme objetivo desta tarefa. Para isso, execute o seguinte comando no console do Mininet:

```
sh ovs-ofctl dump-flows s1
```

A saída esperada é:

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1223.545s, table=0, n_packets=1, n_bytes=98, idle_age=1106, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04 actions=drop
 cookie=0x0, duration=1233.575s, table=0, n_packets=1, n_bytes=98, idle_age=1116, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=1183.474s, table=0, n_packets=1, n_bytes=98, idle_age=1066, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02 actions=drop
 cookie=0x0, duration=1173.45s, table=0, n_packets=1, n_bytes=98, idle_age=1056, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03 actions=drop
 cookie=0x0, duration=1193.456s, table=0, n_packets=1, n_bytes=98, idle_age=1076, in_port=4,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:01 actions=drop
 cookie=0x0, duration=1243.608s, table=0, n_packets=1, n_bytes=98, idle_age=1126, in_port=2,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=drop
 cookie=0x0, duration=1263.69s, table=0, n_packets=9, n_bytes=882, idle_age=1086, in_port=3 actions=output:1
 cookie=0x0, duration=1273.718s, table=0, n_packets=9, n_bytes=882, idle_age=1096, in_port=1 actions=output:3
mininet>
```

Observe que a saída acima contém menos entradas do que era esperado. Quais entradas eram esperadas acima e por que elas não foram inseridas?

Desafio 1: evitar perda de pacotes slowpath

Você deve ter notado nos testes acima que o teste de conectividade entre H1 e H3 resultou na perda de dados nos primeiros pacotes do fluxo. Como sugestão para identificar o problema, recomenda-se repetir o teste porém com a captura de pacotes Wireshark (encerre o Mininet com CTRL+D e inicie novamente). Compare a cadeia de mensagens OF com aquelas mensagens apresentadas na Aula 01.

Dica: pesquise sobre o buffer_id

Desafio 2: encaminhar em modo flood

Observe que na aplicação que desenvolvemos, foi utilizada apenas o encaminhamento da porta 1 para 3 e vice-versa. Lembre-se que existe uma porta especial do OF para encaminhar mensagens em modo flood.

Dica: reveja o uso das condições e simplifique a mensagem forma de encaminhamento para usar o modo flood (port=of.OFPP_ALL).

Prática 04 – Aplicação "single-hub.py" no Ryu

1. O primeiro passo é instalar o Ryu na máquina virtual. Para isso, vamos executar os seguintes comandos para instalar as dependências via pacote debian e via PIP (note que apenas uma linha):

Instalar as dependências via pacotes:

```
sudo apt-get install -y libxslt1-dev  libffi-dev   msgpack-python python-
setuptools python-nose python-pip python-dev
```

Instalar as dependências via PIP:

```
sudo pip install ipaddr networkx bitarray netaddr oslo.config routes webob
paramiko mock xml_compare pyflakes pylint debtcollector oslo.i18n rfc3986
greenlet tinyrpc ovs 'eventlet>=0.18.2,!=0.18.3,!=0.20.1,<0.21.0'
```

2. Agora o Ryu será instalado através dos seguintes comandos:

```
cd ~/
git clone git://github.com/osrg/ryu.git
cd ryu; sudo python ./setup.py install
```

3. O primeiro passo é criar a aplicação "single-hub" e armazenar ela no arquivo ~/single-hub.py. Para isso, devemos executar o seguinte comando para fazer o download da aplicação no local correto:

```
cd ~/
wget http://homes.dcc.ufba.br/~italo/oficina-openflow-2017/single-hub.py
```

2. Abrir um terminal "xterm" e executar o seguinte comando para iniciar o controlador Ryu:

```
ryu-manager ~/single-hub.py
```

Caso tenha executado com sucesso, a saída deve ser:

```
mininet@oficina-openflow-ufba:~$ ryu-manager ~/single-hub.py
loading app /home/mininet/single-hub.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app /home/mininet/single-hub.py of SingleHub
```

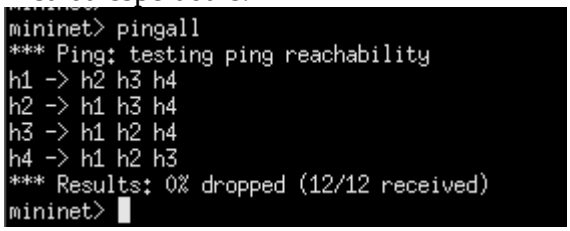
3. Em seguida, vamos abrir um novo terminal "xterm" e iniciar a topologia via Mininet. Para isso, devemos executar o seguinte comando:

```
sudo mn --topo single,4 --mac --arp --controller remote
```

4. Vamos testar a conectividade entre os hosts. Para isso, execute o seguinte comando no console do Mininet:

```
pingall
```

A saída esperada é:



Dica: como estamos trabalhando em modo “hub”, os pacotes são encaminhados para todas as portas. Assim, caso o leitor tenha curiosidade para validar isso, deverá abrir um console xterm no nó h2 e executar um tcpdump, ao passo que também executa um teste de ICMP entre h1 e h3.