SDN

Samara Ribeiro Silva
Instituto Tecnológico de Aeronáutica, Laboratório de Redes de Computadores e Internet (CES-35). Professora Cecilia de Azevedo Castro Cesar, São José dos Campos, São Paulo, 08 de novembro de 2021.

Laboratório número 4

1. Compreendendo o *simple-switch*

Terminal 1

```
mininet@mininet-vm:~/ryu$ sudo mn --topo single,3 --mac --controller remote --switch ovsk,prot
ocols=OpenFlow13
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Terminal 2

```
mininet@mininet-vm:~/ryu$ sudo ./bin/ryu-manager --verbose ryu/app/simple_switch_13.py
loading app ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK SimpleSwitch13
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPSwitchFeatures
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'SimpleSwitch13': {'main'}}
  PROVIDES EventOFPSwitchFeatures TO {'SimpleSwitch13': {'config'}}
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
connected socket:<eventlet.greenio.base.GreenSocket object at 0x7f07e6246dc0> address:('127.0.
0.1', 54946)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x7f07e625ebe0>
move onto config mode
EVENT ofp_event->SimpleSwitch13 EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0xc28ed3e0,OFPSwitchFeatures(auxi
liary_id=0,capabilities=79,datapath_id=1,n_buffers=0,n_tables=254)
move onto main mode
```

- **Testando o ping utilizando o arquivo *simple_switch.py***

Terminal 1	**Terminal 2**

Ping all
`mininet> pingall`
`*** Ping: testing ping reachability`
`h1 -> h2 h3`
`h2 -> h1 h3`
`h3 -> h1 h2`
`*** Results: 0% dropped (6/6 received)`
`mininet>`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:01 2`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:01 3`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:03 1`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:02 ff:ff:ff:ff:ff:ff 2`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3`
	`EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn`
	`packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:03 2`

Observe que todos os pacotes foram recebidos e nenhum foi descartado.

2. “Filhote” de *firewall* (*ffw.py)*

Para a construção do firewall foram realizadas as seguintes modificações no código disponibilizado em *simple_switch.py:*

```
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        ...
        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
            if pkt.get_protocols(icmp.icmp) and out_port == 2:
                self.logger.info("icmp packet to host 2 dropped")
                return
        else:
            out_port = ofproto.OFPP_FLOOD


        ...
        # install a flow to avoid packet_in next time
        # a porta 2 deve ser incluida nas excecoes para garantir que o pacote seja
descartado
        if out_port != ofproto.OFPP_FLOOD and out_port != 2:
            ...
```

Verificou-se se o protocolo do pacote é ICMP e se a porta de destino é de número 2. Caso positivo o pacote é descartado e se as condições não forem satisfeitas o pacote segue normalmente.

- **Testando o ping utilizando o arquivo *ffw.py***

Tabela 1: pingall

Terminal 1

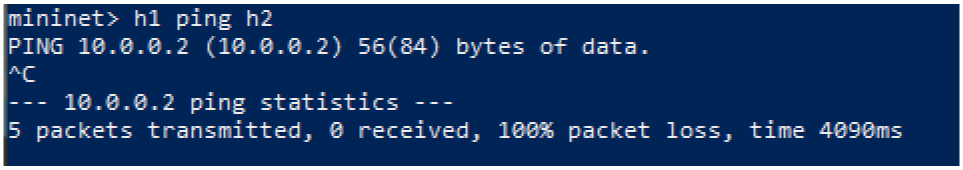
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3
h2 -> X X
h3 -> h1 X
*** Results: 66% dropped (2/6 received)
mininet>
```

Terminal 2

```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:01 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:01 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:03 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:02 ff:ff:ff:ff:ff:ff 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:03 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
```

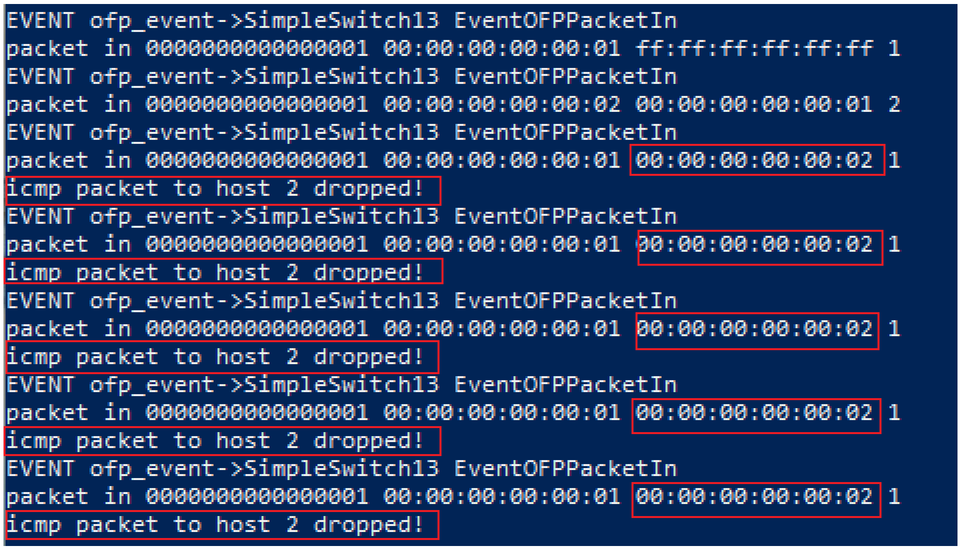
`Tabela 2: h1 ping h2`

Terminal 1



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4090ms
```

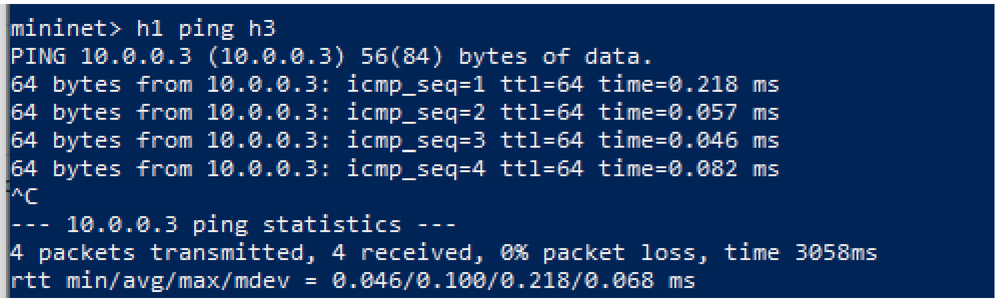
Terminal 2



```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:01 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
```

`Tabela 3: h1 ping h3`

Terminal 1



```
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=1 ttl=64 time=0.218 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.082 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3058ms
rtt min/avg/max/mdev = 0.046/0.100/0.218/0.068 ms
```

Tabela 4: h2 ping h1

Terminal 1

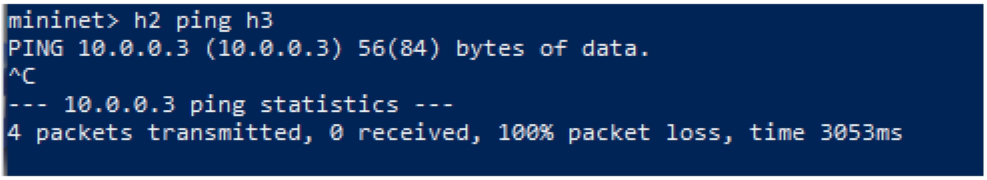
```
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2037ms

mininet>
```

Terminal 2

```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
```

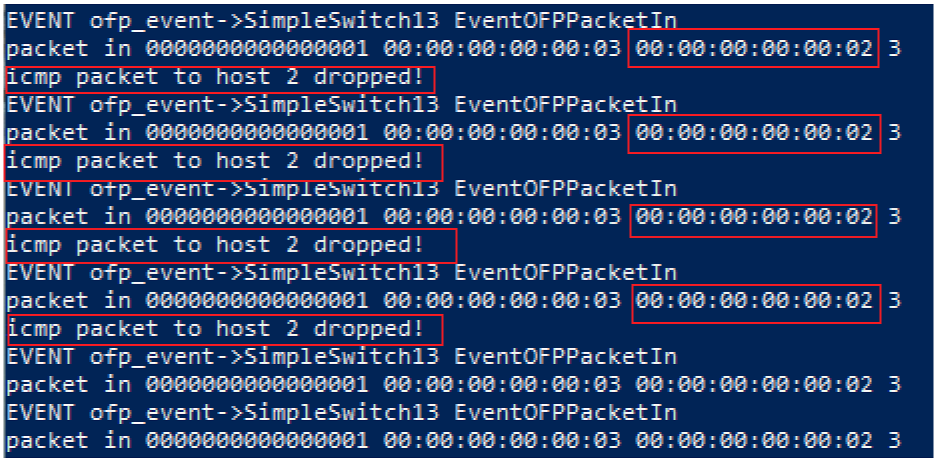
`Tabela 5: h2 ping h3`

Terminal 1



```
mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3053ms
```

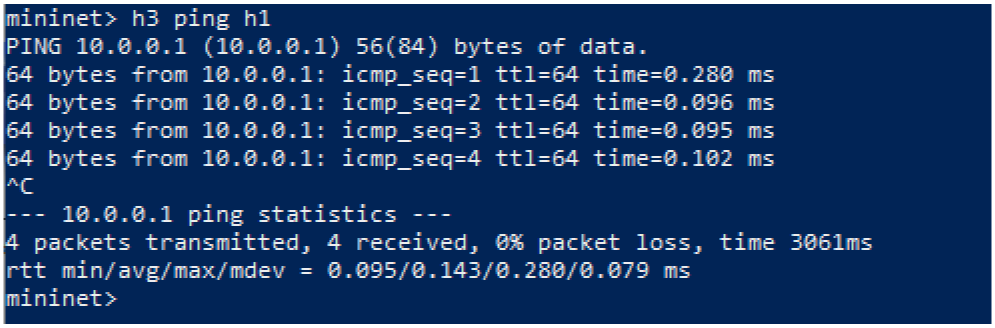
Terminal 2



```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
```

`Tabela 6: h3 ping h1`

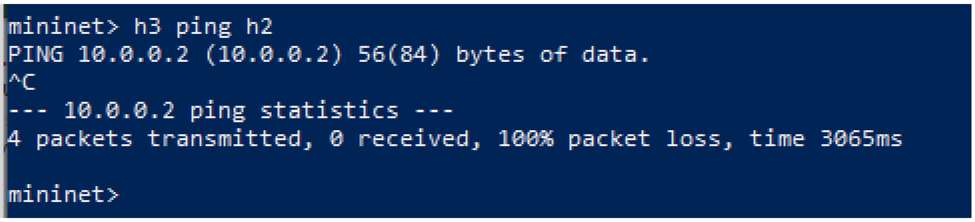
Terminal 1



```
mininet> h3 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.280 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.096 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.102 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.095/0.143/0.280/0.079 ms
mininet>
```

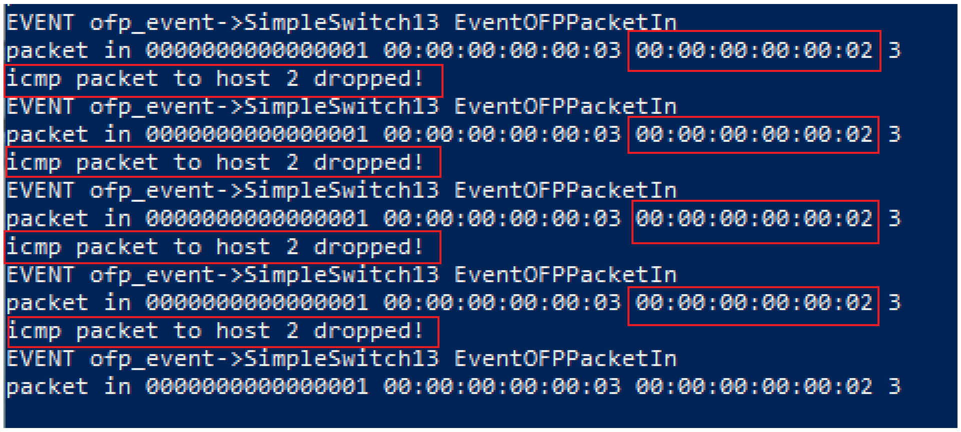
Tabela 7: h3 ping h2

Terminal 1



```
mininet> h3 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3065ms

mininet>
```

Terminal 2



```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
icmp packet to host 2 dropped!
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 00:00:00:00:00:03 00:00:00:00:00:02 3
```

Ao realizar o ping entre h1 e h2, o h1 envia uma mensagem de requisição para h2 e se h2 estiver em condições de responder ele envia uma mensagem de resposta para h1.

Como foi alterado o código para descartar os pacotes do protocolo ICMP com destino ao host 2 espera-se que não seja possível realizar hX ping h2 (pois h2 não receberá a mensagem de request) e nem h2 ping hX (pois h2 não receberá a mensagem de reply). Nas tabelas 1 a 7 é possível observar que o ping entre h1 e h3 é realizado normalmente e ping entre qualquer host e h2 não é completado pois o pacote ICMP é descartado conforme destacado em vermelho nas figuras.

Observe nos registros tcpdump abaixo do ping entre h1 e h3 (marcadas em vermelho), h1 e h2 (marcadas em azul) e por fim entre h2 e h3 (marcadas em amarelo).

```
mininet> h1 ping -c 1 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=13.7 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 13.685/13.685/13.685/0.000 ms
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h2 ping -c 1 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

"Node: h1"

```
root@mininet-vm:~# tcpdump -en -i h1-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:36:19.545067 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.1, length 28
18:36:19.553856 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
18:36:19.553872 00:00:00:00:00:01 > 00:00:00:00:00:03, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.3: ICMP echo request, id 6605, seq 1, length 64
18:36:19.558729 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 6605, seq 1, length 64
18:36:24.809868 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.1 tell 10.0.0.3, length 28
18:36:24.809904 00:00:00:00:00:01 > 00:00:00:00:00:03, ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
18:36:43.773425 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:36:43.781586 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
18:36:43.781597 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 6610, seq 1, length 64
18:37:04.908882 00:00:00:00:00:02 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.2, length 28
```

"Node: h2"

```
root@mininet-vm:~# tcpdump -en -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:36:19.546940 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.1, length 28
18:36:43.775185 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:36:43.775214 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
18:37:04.906412 00:00:00:00:00:02 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.2, length 28
18:37:04.912948 00:00:00:00:00:03 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
18:37:04.912960 00:00:00:00:00:02 > 00:00:00:00:00:03, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.3: ICMP echo request, id 6614, seq 1, length 64
18:37:10.127197 00:00:00:00:00:03 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.3, length 28
18:37:10.127239 00:00:00:00:00:02 > 00:00:00:00:00:03, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
```

"Node: h3"

```
root@mininet-vm:~# tcpdump -en -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:36:19.546941 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.1, length 28
18:36:19.546966 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
18:36:19.558656 00:00:00:00:00:01 > 00:00:00:00:00:03, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.3: ICMP echo request, id 6605, seq 1, length 64
18:36:19.558681 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 6605, seq 1, length 64
18:36:24.808196 00:00:00:00:00:03 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.1 tell 10.0.0.3, length 28
18:36:24.813470 00:00:00:00:00:01 > 00:00:00:00:00:03, ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
18:36:43.775216 00:00:00:00:00:01 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:37:04.908884 00:00:00:00:00:02 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.3 tell 10.0.0.2, length 28
18:37:04.908913 00:00:00:00:00:03 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
18:37:04.916720 00:00:00:00:00:02 > 00:00:00:00:00:03, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.3: ICMP echo request, id 6614, seq 1, length 64
18:37:04.916742 00:00:00:00:00:03 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.3 > 10.0.0.2: ICMP echo reply, id 6614, seq 1, length 64
18:37:10.120163 00:00:00:00:00:03 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.3, length 28
18:37:10.128838 00:00:00:00:00:02 > 00:00:00:00:00:03, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
```

Observe que o ping entre h1 e h3 ocorre normalmente e as mensagens de request e reply são enviadas e recebidas corretamente. Já o ping entre h1 e h2 o pacote de *request* enviado por h1 não é recebido por h2, uma vez que esse recebimento foi bloqueado pelo ffw.py. O ping entre h2 e h3, também, não foi executado com sucesso, pois a mensagem reply enviada pelo h3 não foi recebida pelo h2.

3. Hub

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        ...

        out_port = ofproto.OFPP_FLOOD

        actions = [parser.OFPActionOutput(out_port)]

        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data

        out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                  in_port=in_port, actions=actions, data=data)
        datapath.send_msg(out)
```

Para que os pacotes sejam replicados para todas as portas basta definir $out\_port = ofproto.OFPP\_FLOOD$.

$ofproto.OFPP\_FLOOD$ retorna todas as portas físicas, exceto a porta de entrada e aquelas desabilitadas pelo Spanning Tree Protocol.

```
mininet@mininet-vm: ~/ryu
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=8.72 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=6.51 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=9.19 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 6.509/8.139/9.193/1.169 ms
mininet>
```

```
"Node: h1"
root@mininet-vm:~/ryu# tcpdump -en -i h1-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:04:06.346055 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 1, length 64
18:04:06.354753 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 1, length 64
18:04:07.347707 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 2, length 64
18:04:07.354194 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 2, length 64
18:04:08.349642 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 3, length 64
18:04:08.358810 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 3, length 64
18:04:11.495665 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:04:11.502014 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.1 tell 10.0.0.2, length 28
18:04:11.502049 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
18:04:11.517265 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
```

```
"Node: h2"
root@mininet-vm:~/ryu# tcpdump -en -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:04:06.347809 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 1, length 64
18:04:06.347923 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 1, length 64
18:04:07.349735 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 2, length 64
18:04:07.349756 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 2, length 64
18:04:08.353868 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 3, length 64
18:04:08.353902 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 3, length 64
18:04:11.495646 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.1 tell 10.0.0.2, length 28
18:04:11.506314 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:04:11.506328 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
18:04:11.513250 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
```

```
"Node: h3"
root@mininet-vm:~/ryu# tcpdump -en -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:04:06.347926 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 1, length 64
18:04:06.354755 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 1, length 64
18:04:07.349736 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 2, length 64
18:04:07.354197 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 2, length 64
18:04:08.353870 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5829, seq 3, length 64
18:04:08.358812 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5829, seq 3, length 64
18:04:11.502056 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.1 tell 10.0.0.2, length 28
18:04:11.506331 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Request who-has 10.0.0.2 tell 10.0.0.1, length 28
18:04:11.513266 00:00:00:00:00:01 > 00:00:00:00:00:02, ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
18:04:11.517279 00:00:00:00:00:02 > 00:00:00:00:00:01, ethertype ARP (0x0806), length 42: Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
```

Observe que o host h3 também recebeu as mensagens de *request* do h1 e de *reply* do h2, mas não enviou nenhuma mensagem porque nenhuma das mensagens recebidas era endereçada para ele.