Relatório do EP3
MAC0352 – Redes de Computadores e Sistemas Distribuídos – 1/2021

Arthur Font Gouveia (12036152)

26/07/2021

1 Passo 0

Na definição do protocolo OpenFlow, o que um switch faz toda vez que ele recebe um pacote que ele nunca recebeu antes?

R: De acordo com o protocolo OpenFlow, quando o switch recebe um pacote de um novo endereço, este pacote é encaminhado ao controlador OpenFlow, que é responsável por modificar o switch para que o mesmo possa atender a este novo fluxo.

2 Passo 2

Com o acesso à Internet funcionando em sua rede local, instale na VM o programa `traceroute` usando `sudo apt install traceroute` e escreva abaixo a saída do comando `sudo traceroute -I www.inria.fr`. Pela saída do comando, a partir de qual salto os pacotes alcançaram um roteador na Europa? Como você chegou a essa conclusão?

R: Os pacotes alcançaram um roteador na Europa a partir do oitavo salto. Foi possível observar através de um aumento significativo no tempo de resposta (*round trip-time*) e confirmado através de uma ferramenta de *IP lookup*.

```
traceroute to www.inria.fr (128.93.162.83), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.127 ms  0.090 ms  0.069 ms
 2  Broadcom.Home (192.168.15.1)  2.713 ms  2.718 ms  8.205 ms
 3  * * *
 4  187-100-186-58.dsl.telesp.net.br (187.100.186.58)  8.818 ms  9.783 ms  9.801 ms
 5  152-255-140-51.user.vivozap.com.br (152.255.140.51)  12.089 ms  12.138 ms  12.139 ms
 6  213.99.17.198 (213.99.17.198)  14.357 ms  9.165 ms  9.271 ms
 7  * * *
 8  94.142.118.184 (94.142.118.184)  117.294 ms  118.007 ms *
 9  ae5.cr6-mia1.ip4.gtt.net (208.116.240.149)  123.972 ms  124.438 ms  124.439 ms
10  et-3-3-0.cr2-par7.ip4.gtt.net (213.200.119.214)  206.099 ms  206.520 ms  206.464 ms
11  renater-gw-ix1.gtt.net (77.67.123.206)  240.331 ms  236.375 ms  236.002 ms
12  te1-1-inria-rtr-021.noc.renater.fr (193.51.177.107)  220.395 ms  221.012 ms  222.212 ms
13  inria-rocquencourt-gi3-2-inria-rtr-021.noc.renater.fr (193.51.184.177)  204.663 ms  204.652 ms  205.279 ms
14  unit240-reth1-vfw-ext-dc1.inria.fr (192.93.122.19)  238.064 ms  238.268 ms  238.414 ms
15  www.inria.fr (128.93.162.83)  219.997 ms  218.185 ms  218.602 ms
```

3 Passo 3 - Parte 1

Execute o comando `iperf`, conforme descrito no tutorial, antes de usar a opção `--switch user`, 5 vezes. Escreva abaixo o valor médio e o intervalo de confiança da taxa retornada (considere sempre o primeiro valor do vetor retornado).

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['18.4 Gbits/sec', '18.5 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['19.4 Gbits/sec', '19.4 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['19.2 Gbits/sec', '19.2 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['19.2 Gbits/sec', '19.2 Gbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['18.2 Gbits/sec', '18.2 Gbits/sec']
```

$\overline{X}$ = 18.88 Gbits/sec
IC de 95%: [18,4208; 19.3392] Gbits/sec

4 Passo 3 - Parte 2

Execute o comando `iperf`, conforme descrito no tutorial, com a opção `--switch user`, 5 vezes. Escreva abaixo o valor médio e o intervalo de confiança da taxa retornada (considere sempre o primeiro valor do vetor retornado). O resultado agora corresponde a quantas vezes menos o da Seção anterior? Qual o motivo dessa diferença?

```
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['391 Mbits/sec', '393 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['401 Mbits/sec', '403 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['404 Mbits/sec', '405 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['391 Mbits/sec', '392 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['336 Mbits/sec', '338 Mbits/sec']
```

$\overline{X}$ = 384.6 Mbits/sec
IC de 95%: [360.9864; 408.2135] Mbits/sec

R: O resultado corresponde a aproximadamente 49 vezes menos. O motivo desta diferença é que, com o user-space switch, os pacotes precisam passar do user-space para o kernel-space e do kernel-space para o user-space, o que diminui consideralvelmente a taxa de transmissão. Já na seção anterior, os pacotes permaneciam dentro do kernel-space antes de ir diretamente ao *switch*.

5 Passo 4 - Parte 1

Execute o comando `iperf`, conforme descrito no tutorial, usando o controlador `of_tutorial.py` original sem modificação, 5 vezes. Escreva abaixo o valor médio e o intervalo de confiança da taxa retornada (considere sempre o primeiro valor do vetor retornado). O resultado agora corresponde a quantas vezes menos o da Seção 3? Qual o motivo para essa diferença? Use a saída do comando `tcpdump`, deixando claro em quais computadores virtuais ele foi executado, para justificar a sua resposta.

$\overline{X}$ = 26.48 Mbits/sec

IC de 95%: [24.2992; 28.6607] Mbits/sec

R: O resultado corresponde a aproximadamente 713 vezes menos. O motivo desta diferença é porque os pacotes estão sendo enviados ao controlador, que está atuando como um *hub*. Portanto, todos os pacotes estão sendo enviados a todos os *hosts* conectados ao controlador. Para chegar a esta conclusão, foram excecutados os seguintes comandos nas máquinas virtuais $h1$, $h2$ e $h3$ e suas respectivas saídas foram:

$h1$:

```
$ ping -c1 10.0.0.2
```

$h2$:

```
$ tcpdump -XX -n -i h2-eth0
09:47:54.429435 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
09:47:54.429471 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
09:47:54.431589 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 10352, seq 1, length 64
0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
0x0010:  0054 cf81 4000 4001 5725 0a00 0001 0a00  .T..@.@.W%......
0x0020:  0002 0800 97df 2870 0001 3a44 fc60 0000  ......(p..:D.`..
0x0030:  0000 3c37 0600 0000 0000 1011 1213 1415  ..<7............
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
09:47:54.431615 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 10352, seq 1, length 64
0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
0x0010:  0054 3eb5 0000 4001 27f2 0a00 0002 0a00  .T>...@.'.......
0x0020:  0001 0000 9fdf 2870 0001 3a44 fc60 0000  ......(p..:D.`..
0x0030:  0000 3c37 0600 0000 0000 1011 1213 1415  ..<7............
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
09:47:59.446840 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0000 0a00 0001                 ..........
09:47:59.491016 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0002 0a00 0002                 ..........
```

$h3$:

```
$ tcpdump -XX -n -i h3-eth0
09:47:54.429433 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
09:47:54.430512 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
09:47:54.431587 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 10352, seq 1, length 64
0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
0x0010:  0054 cf81 4000 4001 5725 0a00 0001 0a00  .T..@.@.W%......
0x0020:  0002 0800 97df 2870 0001 3a44 fc60 0000  ......(p..:D.`..
0x0030:  0000 3c37 0600 0000 0000 1011 1213 1415  ..<7............
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
09:47:54.432447 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 10352, seq 1, length 64
0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
0x0010:  0054 3eb5 0000 4001 27f2 0a00 0002 0a00  .T>...@.'.......
0x0020:  0001 0000 9fdf 2870 0001 3a44 fc60 0000  ......(p..:D.`..
0x0030:  0000 3c37 0600 0000 0000 1011 1213 1415  ..<7............
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
09:47:59.488469 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0000 0a00 0001                 ..........
09:47:59.491011 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0002 0a00 0002                 ..........
```

6 Passo 4 - Parte 2

Execute o comando `iperf`, conforme descrito no tutorial, usando o seu controlador `switch.py`, 5 vezes. Escreva abaixo o valor médio e o intervalo de confiança da taxa retornada (considere sempre o primeiro valor do vetor retornado). O resultado agora corresponde a quantas vezes mais o da Seção anterior? Qual o motivo dessa diferença? Use a saída do comando `tcpdump`, deixando claro em quais computadores virtuais ele foi executado, para justificar a sua resposta.

$\overline{X}$ = 29.7 Mbits/sec

IC de 95%: [28.2543; 31.1457] Mbits/sec

R: O resultado corresponde a aproximadamente 1,12 vezes mais o da seção anterior. O motivo desta diferença é porque agora o controlador está atuando como um *switch*. Portanto, os pacotes estão sendo enviados ao(s) *hosts* correspondente(s), o que melhora a taxa de trasnmissão. Pode-se comprovar através da excecução dos seguintes comandos nas máquinas virtuais $h1$, $h2$ e $h3$:

$h1$:

```
$ ping -c1 10.0.0.2
```

$h2$:

```
$ tcpdump -XX -n -i h2-eth0
10:54:37.698117 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
10:54:37.698169 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
10:54:37.701009 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 3344, seq 1, length 64
0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
0x0010:  0054 a566 4000 4001 8140 0a00 0001 0a00  .T.f@.@..@......
0x0020:  0002 0800 9d63 0d10 0001 dd53 fc60 0000  .....c.....S.`..
0x0030:  0000 ab03 0a00 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
10:54:37.701038 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 3344, seq 1, length 64
0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
0x0010:  0054 fc18 0000 4001 6a8e 0a00 0002 0a00  .T....@.j.......
0x0020:  0001 0000 a563 0d10 0001 dd53 fc60 0000  .....c.....S.`..
0x0030:  0000 ab03 0a00 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
10:54:42.712565 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0000 0a00 0001                 ..........
10:54:42.716068 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0002 0a00 0002                 ..........
```

$h3$:

```
$ tcpdump -XX -n -i h3-eth0
10:54:37.698113 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
```

Como é possível observar, desta vez o $h3$ recebeu apenas o pacote de $broadcast$ para identificação do IP de destino dos pacotes enviados pelo comando `ping`.

7 Passo 4 - Parte 3

Execute o comando `iperf`, conforme descrito no tutorial, usando o seu controlador `switch.py` melhorado, 5 vezes. Escreva abaixo o valor médio e o intervalo de confiança da taxa retornada (considere sempre o primeiro valor do vetor retornado). O resultado agora corresponde a quantas vezes mais o da Seção anterior? Qual o motivo dessa diferença? Use a saída do comando `tcpdump`, deixando claro em quais computadores virtuais ele foi executado, e saídas do comando `sudo ovs-ofctl`, com os devidos parâmetros, para justificar a sua resposta.

$\overline{X}$ = 18.66 Gbits/sec

IC de 95%: [18.0351; 19.2848] Gbits/sec

R: O resultado corresponde a aproximadamente 628 vezes mais o da seção anterior. O motivo desta grande diferença é porque agora o *switch* reconhece um fluxo antigo através das regras do controlador. Logo os pacotes trafegam diretamente à porta conhecida, exceto os pacotes iniciais. Desta vez, para testar se o switch reconhce um fluxo conhecido, foi necesário a execução de dois comandos `ping`. Os comandos e suas respectivas saídas estão representados abaixo (exceto a saída do comando `ping`):

Primeiro `ping`:

$h1$:

```
$ ping -c1 10.0.0.2
```

$h2$:

```
$ tcpdump -XX -n -i h2-eth0
20:15:40.018126 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
20:15:40.018187 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
20:15:40.027544 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 8913, seq 1, length 64
0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
0x0010:  0054 39d7 4000 4001 eccf 0a00 0001 0a00  .T9.@.@.........
0x0020:  0002 0800 a213 22d1 0001 5cd7 fc60 0000  ......"...\..`..
0x0030:  0000 1b0f 0000 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
20:15:40.027565 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 8913, seq 1, length 64
0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
0x0010:  0054 3f6d 0000 4001 273a 0a00 0002 0a00  .T?m..@.':......
0x0020:  0001 0000 aa13 22d1 0001 5cd7 fc60 0000  ......"...\..`..
0x0030:  0000 1b0f 0000 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
20:15:45.029164 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0000 0a00 0001                 ..........
20:15:45.044183 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0002 0a00 0002                 ..........
```

$h3$:

```
$ tcpdump -XX -n -i h3-eth0
20:15:40.018121 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
```

Comando *ovs-ofctl* em um terminal SSH:

```
$ sudo ovs-ofctl dump-flows s1
cookie=0x0, duration=9.527s, table=0, n_packets=1, n_bytes=98, idle_age=9, icmp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.1,
 nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:2
cookie=0x0, duration=9.525s, table=0, n_packets=1, n_bytes=98, idle_age=9, icmp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.0.2,
 nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:1
cookie=0x0, duration=4.51s, table=0, n_packets=1, n_bytes=42, idle_age=4, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.1,
 arp_tpa=10.0.0.2,arp_op=2 actions=output:2
cookie=0x0, duration=9.532s, table=0, n_packets=1, n_bytes=42, idle_age=9, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
 arp_tpa=10.0.0.1,arp_op=2 actions=output:1
cookie=0x0, duration=4.515s, table=0, n_packets=1, n_bytes=42, idle_age=4, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
 arp_tpa=10.0.0.1,arp_op=1 actions=output:1
```

Segundo `ping`:
$h1$:

```
$ ping -c1 10.0.0.2
```

$h2$:

```
$ tcpdump -XX -n -i h2-eth0
20:15:40.018121 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........

mininet@mininet-vm:~$ cat a2.txt
20:15:40.018126 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0002                 ..........
20:15:40.018187 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
20:15:40.027544 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 8913, seq 1, length 64
0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
0x0010:  0054 39d7 4000 4001 eccf 0a00 0001 0a00  .T9.@.@.........
0x0020:  0002 0800 a213 22d1 0001 5cd7 fc60 0000  ......"...\..`..
0x0030:  0000 1b0f 0000 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
20:15:40.027565 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 8913, seq 1, length 64
0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
0x0010:  0054 3f6d 0000 4001 273a 0a00 0002 0a00  .T?m..@.':......
0x0020:  0001 0000 aa13 22d1 0001 5cd7 fc60 0000  ......"...\..`..
0x0030:  0000 1b0f 0000 0000 0000 1011 1213 1415  ................
0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
0x0060:  3637                                     67
```

```
20:15:45.029164 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
0x0020:  0000 0000 0000 0a00 0001                 ..........
20:15:45.044183 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0002 0a00 0002                 ..........
```

$h3$:

```
$ tcpdump -XX -n -i h3-eth0
```

Comando *ovs-ofctl* em um terminal SSH:

```
$ sudo ovs-ofctl dump-flows s1
cookie=0x0, duration=59.585s, table=0, n_packets=2, n_bytes=196, idle_age=13, icmp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.1,
 nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:2
cookie=0x0, duration=59.583s, table=0, n_packets=2, n_bytes=196, idle_age=13, icmp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.0.2,
 nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:1
cookie=0x0, duration=54.568s, table=0, n_packets=2, n_bytes=84, idle_age=8, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.1,
 arp_tpa=10.0.0.2,arp_op=2 actions=output:2
cookie=0x0, duration=8.419s, table=0, n_packets=1, n_bytes=42, idle_age=8, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.1,
 arp_tpa=10.0.0.2,arp_op=1 actions=output:2
cookie=0x0, duration=59.59s, table=0, n_packets=2, n_bytes=84, idle_age=8, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
 arp_tpa=10.0.0.1,arp_op=2 actions=output:1
cookie=0x0, duration=54.573s, table=0, n_packets=2, n_bytes=84, idle_age=8, arp,
 vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
 arp_tpa=10.0.0.1,arp_op=1 actions=output:1
```

No primeiro `ping`, assim como na seção anterior, $h3$ recebeu apenas o pacote de $broadcast$ para identificação do IP de destino. Porém, é possível observar que o segundo `ping` não chegou ao $h3$. Como pode-se observar na saída do comando*ovs-ofctl*, o número de bytes duplicou em cada fluxo após a execução do segundo `ping`, o que revela a reutilização dos fluxos previamente estabelecidos.

8 Passo 5

Explique a lógica implementada no seu controlador `firewall.py` e mostre saídas de comandos que comprovem que ele está de fato funcionando (saídas dos comandos `tcpdump`, `sudo ovs-ofctl`, `nc`, `iperf` e `telnet` são recomendadas)

R: O controlador `firewall.py` verifica se cada pacote enviado coincide inteiramente com alguma regra definida no arquivo `rules.json`. A lógica implementada consiste em um *loop* que verifica para cada regra definida, se todos os paramêtros especificados nesta regra coincidem com os parâmetros do pacote que está sendo enviado. Caso alguma regra coincida, este pacote é barrado pelo controlador e não é enviado ao IP de destino.

Para testar o funcionamento do controlador foram utilizados os comandos `pingall`, `iperf` e `tcpdump` e o seguinte arquivo `rules.json`:

```
{
    "0" : {
        "src_ip": "10.0.0.1",
        "dst_ip": "10.0.0.2"
    },
    "1" : {
        "protocol": "UDP"
    }
}
```

Para testar a primeira regra, foi executado o seguinte comando e sua respectiva saída está representada abaixo:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3
h2 -> X h3
h3 -> h1 h2
*** Results: 33% dropped (4/6 received)
```

Como esperado, os pacotes entre os hosts $h1$ e $h2$ foram descartados pelo controlador, já os demais pacotes foram entregues.

Para testar a segunda regra, foram executados os seguintes comandos e sua respectivas saídas estão representadas abaixo:

$h1$:

```
$ iperf -c 10.0.0.3 -u
------------------------------------------------------------
Client connecting to 10.0.0.3, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 15] local 10.0.0.1 port 59425 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 15]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 15] Sent 893 datagrams
```

$h3$:

```
$ tcpdump -XX -n -i h3-eth0
19:58:08.966952 ARP, Request who-has 10.0.0.3 tell 10.0.0.1, length 28
0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
0x0020:  0000 0000 0000 0a00 0003                 ..........
19:58:08.966976 ARP, Reply 10.0.0.3 is-at 00:00:00:00:00:03, length 28
0x0000:  0000 0000 0001 0000 0000 0003 0806 0001  ................
0x0010:  0800 0604 0002 0000 0000 0003 0a00 0003  ................
0x0020:  0000 0000 0001 0a00 0001                 ..........
```

Através da saída dos comandos executados, pode-se observar que os pacotes entre os hosts $h1$ e $h3$ utilizando o protocolo UDP foram descartados pelo controlador. Apenas o pacote ARP foi entregue ao host $h3$. Portanto, o firewall atuou como o esperado em ambos testes.

9 Configuração dos computadores virtual e real usados nas medições (se foi usado mais de um, especifique qual passo foi feito com cada um)

Computador real:

- OS: Ubuntu 20.04.2 LTS 64-bit
- Processador: Intel® Core™ i5-3317U CPU @ 1.70GHz × 4
- RAM: 4GiB SODIMM DDR3 1333 MHz (0,8 ns)

Computador virtual (VM):

- OS: Ubuntu 14.04.4 LTS 64-bit
- Processador: Intel® Core™ i5-3317U CPU @ 1.70GHz
- RAM: 1024MB

10 Referências

- `https://docs.python.org/pt-br/3/library/json.html`
- `https://openflow.stanford.edu/display/ONL/POX+Wiki.html`
- `https://github.com/Lafaiet/POXDoc/blob/master/PT_BR/Main.md`