Practica SDN

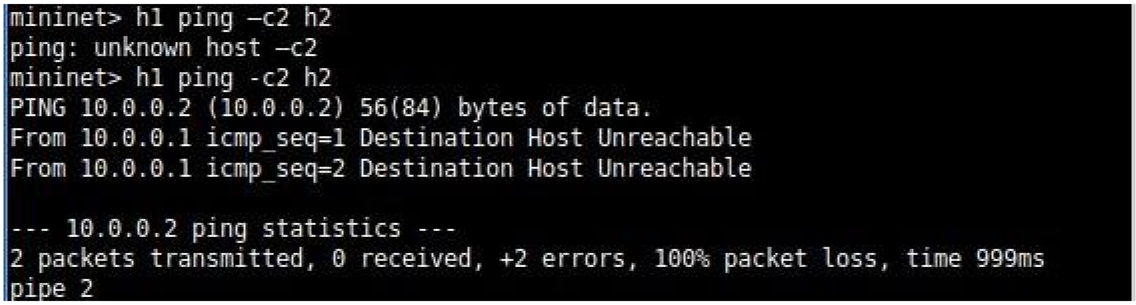
Alumno : Nicolas Bousquet

Practica 1:

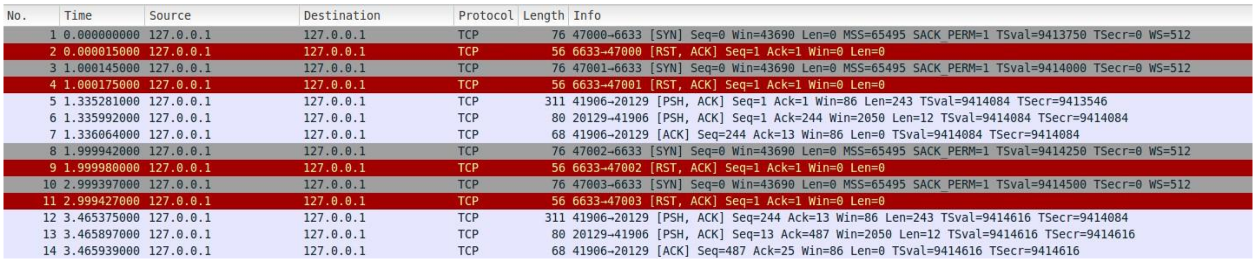
Practica 1A:

Antes de arrancar la aplicación con el controlador Pox o ¿Hay conectividad entre hosts? ¿Por qué ocurre esto?

No hay conectividad entre los hosts porque las reglas del switch no han sido inicializadas por el controlador.



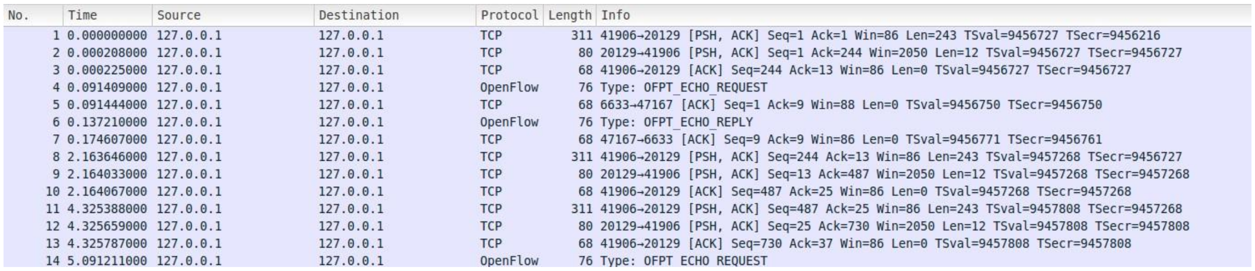
```
mininet> h1 ping –c2 h2
ping: unknown host –c2
mininet> h1 ping -c2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 999ms
pipe 2
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	47000→6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=9413750 TSecr=0 WS=512
2	0.000015000	127.0.0.1	127.0.0.1	TCP	56	6633→47000 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	1.000145000	127.0.0.1	127.0.0.1	TCP	76	47001→6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=9414000 TSecr=0 WS=512
4	1.000175000	127.0.0.1	127.0.0.1	TCP	56	6633→47001 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	1.335281000	127.0.0.1	127.0.0.1	TCP	311	41906→20129 [PSH, ACK] Seq=1 Ack=1 Win=86 Len=243 TSval=9414084 TSecr=9413546
6	1.335992000	127.0.0.1	127.0.0.1	TCP	80	20129→41906 [PSH, ACK] Seq=1 Ack=244 Win=2050 Len=12 TSval=9414084 TSecr=9414084
7	1.336064000	127.0.0.1	127.0.0.1	TCP	68	41906→20129 [ACK] Seq=244 Ack=13 Win=86 Len=0 TSval=9414084 TSecr=9414084
8	1.999942000	127.0.0.1	127.0.0.1	TCP	76	47002→6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=9414250 TSecr=0 WS=512
9	1.999980000	127.0.0.1	127.0.0.1	TCP	56	6633→47002 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10	2.999397000	127.0.0.1	127.0.0.1	TCP	76	47003→6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=9414500 TSecr=0 WS=512
11	2.999427000	127.0.0.1	127.0.0.1	TCP	56	6633→47003 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
12	3.465375000	127.0.0.1	127.0.0.1	TCP	311	41906→20129 [PSH, ACK] Seq=244 Ack=13 Win=86 Len=243 TSval=9414616 TSecr=9414084
13	3.465897000	127.0.0.1	127.0.0.1	TCP	80	20129→41906 [PSH, ACK] Seq=13 Ack=487 Win=2050 Len=12 TSval=9414616 TSecr=9414616
14	3.465939000	127.0.0.1	127.0.0.1	TCP	68	41906→20129 [ACK] Seq=487 Ack=25 Win=86 Len=0 TSval=9414616 TSecr=9414616

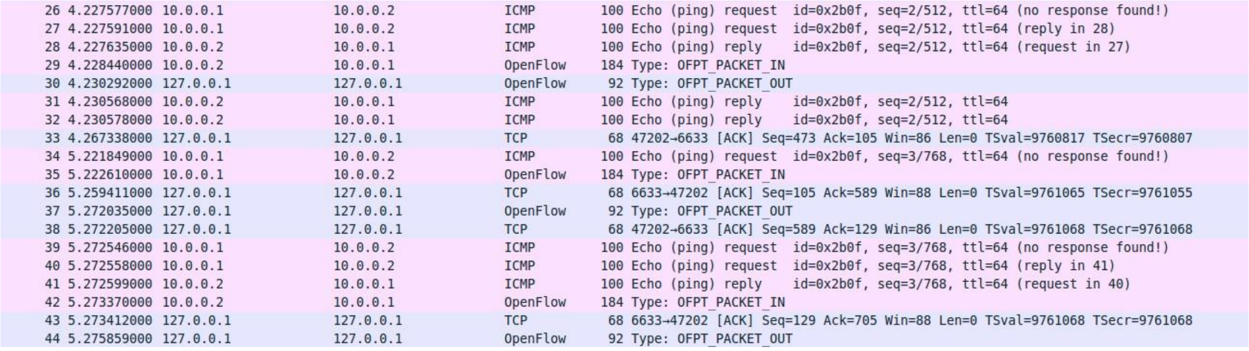
Ejecutar la aplicación forwarding.tutorial_l2_hub o ¿Qué nuevos mensajes aparecen en la red? o Ejecutar un ping entre 2 hosts e indicar que nuevo intercambio de mensajes ocurre en la red y entre que entidades se produce dicho intercambio.

Vemos la aparición de mensajes OpenFlow (ECHO_REQUEST/REPLY) entre el switch y el controlador.



```
No.   Time        Source     Destination  Protocol Length Info
    1 0.000000000 127.0.0.1  127.0.0.1    TCP        311 41906→20129 [PSH, ACK] Seq=1 Ack=1 Win=86 Len=243 TSval=9456727 TSecr=9456216
    2 0.000208000 127.0.0.1  127.0.0.1    TCP         80 20129→41906 [PSH, ACK] Seq=1 Ack=244 Win=2050 Len=12 TSval=9456727 TSecr=9456727
    3 0.000225000 127.0.0.1  127.0.0.1    TCP         68 41906→20129 [ACK] Seq=244 Ack=13 Win=86 Len=0 TSval=9456727 TSecr=9456727
    4 0.091409000 127.0.0.1  127.0.0.1    OpenFlow    76 Type: OFPT_ECHO_REQUEST
    5 0.091444000 127.0.0.1  127.0.0.1    TCP         68 6633→47167 [ACK] Seq=1 Ack=9 Win=88 Len=0 TSval=9456750 TSecr=9456750
    6 0.137210000 127.0.0.1  127.0.0.1    OpenFlow    76 Type: OFPT_ECHO_REPLY
    7 0.174607000 127.0.0.1  127.0.0.1    TCP         68 47167→6633 [ACK] Seq=9 Ack=9 Win=86 Len=0 TSval=9456771 TSecr=9456761
    8 2.163646000 127.0.0.1  127.0.0.1    TCP        311 41906→20129 [PSH, ACK] Seq=244 Ack=13 Win=86 Len=243 TSval=9457268 TSecr=9456727
    9 2.164033000 127.0.0.1  127.0.0.1    TCP         80 20129→41906 [PSH, ACK] Seq=13 Ack=487 Win=2050 Len=12 TSval=9457268 TSecr=9457268
   10 2.164067000 127.0.0.1  127.0.0.1    TCP         68 41906→20129 [ACK] Seq=487 Ack=25 Win=86 Len=0 TSval=9457268 TSecr=9457268
   11 4.325388000 127.0.0.1  127.0.0.1    TCP        311 41906→20129 [PSH, ACK] Seq=487 Ack=25 Win=86 Len=243 TSval=9457808 TSecr=9457268
   12 4.325659000 127.0.0.1  127.0.0.1    TCP         80 20129→41906 [PSH, ACK] Seq=25 Ack=730 Win=2050 Len=12 TSval=9457808 TSecr=9457808
   13 4.325787000 127.0.0.1  127.0.0.1    TCP         68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=9457808 TSecr=9457808
   14 5.091211000 127.0.0.1  127.0.0.1    OpenFlow    76 Type: OFPT_ECHO_REQUEST
```

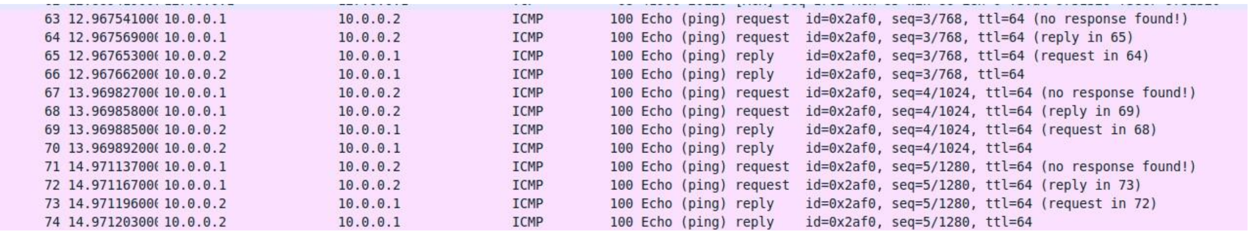
Después de ejecutar forwarding.tutorial_l2_hub.py, aparecen los mensajes PACKET_IN/OUT entre el switch y el controlador.



```
   26 4.227577000 10.0.0.1   10.0.0.2     ICMP       100 Echo (ping) request  id=0x2b0f, seq=2/512, ttl=64 (no response found!)
   27 4.227591000 10.0.0.1   10.0.0.2     ICMP       100 Echo (ping) request  id=0x2b0f, seq=2/512, ttl=64 (reply in 28)
   28 4.227635000 10.0.0.2   10.0.0.1     ICMP       100 Echo (ping) reply    id=0x2b0f, seq=2/512, ttl=64 (request in 27)
   29 4.228440000 10.0.0.2   10.0.0.1     OpenFlow   184 Type: OFPT_PACKET_IN
   30 4.230292000 127.0.0.1  127.0.0.1    OpenFlow    92 Type: OFPT_PACKET_OUT
   31 4.230568000 10.0.0.2   10.0.0.1     ICMP       100 Echo (ping) reply    id=0x2b0f, seq=2/512, ttl=64
   32 4.230578000 10.0.0.2   10.0.0.1     ICMP       100 Echo (ping) reply    id=0x2b0f, seq=2/512, ttl=64
   33 4.267338000 127.0.0.1  127.0.0.1    TCP         68 47202→6633 [ACK] Seq=473 Ack=105 Win=86 Len=0 TSval=9760817 TSecr=9760807
   34 5.221849000 10.0.0.1   10.0.0.2     ICMP       100 Echo (ping) request  id=0x2b0f, seq=3/768, ttl=64 (no response found!)
   35 5.222610000 10.0.0.1   10.0.0.2     OpenFlow   184 Type: OFPT_PACKET_IN
   36 5.259411000 127.0.0.1  127.0.0.1    TCP         68 6633→47202 [ACK] Seq=105 Ack=589 Win=88 Len=0 TSval=9761065 TSecr=9761055
   37 5.272035000 127.0.0.1  127.0.0.1    OpenFlow    92 Type: OFPT_PACKET_OUT
   38 5.272205000 127.0.0.1  127.0.0.1    TCP         68 47202→6633 [ACK] Seq=589 Ack=129 Win=86 Len=0 TSval=9761068 TSecr=9761068
   39 5.272546000 10.0.0.1   10.0.0.2     ICMP       100 Echo (ping) request  id=0x2b0f, seq=3/768, ttl=64 (no response found!)
   40 5.272558000 10.0.0.1   10.0.0.2     ICMP       100 Echo (ping) request  id=0x2b0f, seq=3/768, ttl=64 (reply in 41)
   41 5.272599000 10.0.0.2   10.0.0.1     ICMP       100 Echo (ping) reply    id=0x2b0f, seq=3/768, ttl=64 (request in 40)
   42 5.273370000 10.0.0.2   10.0.0.1     OpenFlow   184 Type: OFPT_PACKET_IN
   43 5.273412000 127.0.0.1  127.0.0.1    TCP         68 6633→47202 [ACK] Seq=129 Ack=705 Win=88 Len=0 TSval=9761068 TSecr=9761068
   44 5.275859000 127.0.0.1  127.0.0.1    OpenFlow    92 Type: OFPT_PACKET_OUT
```

Ejecutar la aplicación forwarding.l2_learning o Ejecutar un ping entre 2 hosts, ¿cuál es la principal diferencia que existe entre esta aplicación y la anteriormente ejecutada? Explicar esta diferencia basándose en los mensajes intercambiados actualmente en la red.

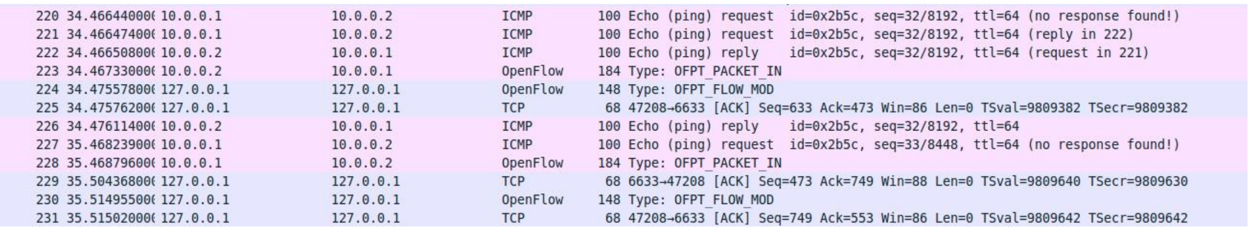
La aplicación forwarding.l2.learning instala las reglas a nivel del switch, por lo que éste no reenvía sistemáticamente los paquetes al controlador (no hay PACKET_IN/OUT sistemático). Sólo lo hace cuando no tiene una regla definida para un paquete.



```
63 12.96754100( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=3/768, ttl=64 (no response found!)
64 12.96756900( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=3/768, ttl=64 (reply in 65)
65 12.96765300( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=3/768, ttl=64 (request in 64)
66 12.96766200( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=3/768, ttl=64
67 13.96982700( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=4/1024, ttl=64 (no response found!)
68 13.96985800( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=4/1024, ttl=64 (reply in 69)
69 13.96988500( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=4/1024, ttl=64 (request in 68)
70 13.96989200( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=4/1024, ttl=64
71 14.97113700( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=5/1280, ttl=64 (no response found!)
72 14.97116700( 10.0.0.1          10.0.0.2          ICMP       100 Echo (ping) request  id=0x2af0, seq=5/1280, ttl=64 (reply in 73)
73 14.97119600( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=5/1280, ttl=64 (request in 72)
74 14.97120300( 10.0.0.2          10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2af0, seq=5/1280, ttl=64
```

¿Qué ocurre si el ping se ejecuta durante un periodo de tiempo lo suficientemente grande (>30seg)? ¿Está reflejado de alguna forma en la captura de Wireshark?

El conmutador reinicia su regla y devuelve un PACKET_IN al controlador.

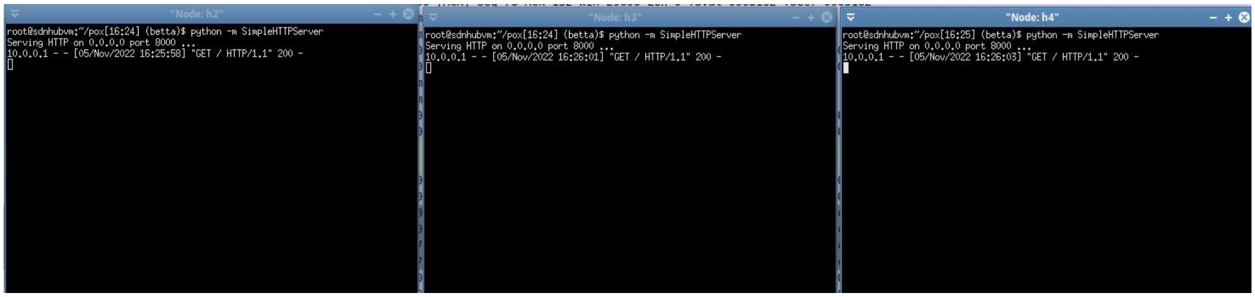


```
220 34.46644000( 10.0.0.1         10.0.0.2          ICMP       100 Echo (ping) request  id=0x2b5c, seq=32/8192, ttl=64 (no response found!)
221 34.46647400( 10.0.0.1         10.0.0.2          ICMP       100 Echo (ping) request  id=0x2b5c, seq=32/8192, ttl=64 (reply in 222)
222 34.46650800( 10.0.0.2         10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2b5c, seq=32/8192, ttl=64 (request in 221)
223 34.46733000( 10.0.0.2         10.0.0.1          OpenFlow   184 Type: OFPT_PACKET_IN
224 34.47557800( 127.0.0.1        127.0.0.1         OpenFlow   148 Type: OFPT_FLOW_MOD
225 34.47576200( 127.0.0.1        127.0.0.1         TCP         68 47208→6633 [ACK] Seq=633 Ack=473 Win=86 Len=0 TSval=9809382 TSecr=9809382
226 34.47611400( 10.0.0.2         10.0.0.1          ICMP       100 Echo (ping) reply    id=0x2b5c, seq=32/8192, ttl=64
227 35.46823900( 10.0.0.1         10.0.0.2          ICMP       100 Echo (ping) request  id=0x2b5c, seq=33/8448, ttl=64 (no response found!)
228 35.46879600( 10.0.0.1         10.0.0.2          OpenFlow   184 Type: OFPT_PACKET_IN
229 35.50436800( 127.0.0.1        127.0.0.1         TCP         68 6633→47208 [ACK] Seq=473 Ack=749 Win=88 Len=0 TSval=9809640 TSecr=9809630
230 35.51495500( 127.0.0.1        127.0.0.1         OpenFlow   148 Type: OFPT_FLOW_MOD
231 35.51502000( 127.0.0.1        127.0.0.1         TCP         68 47208→6633 [ACK] Seq=749 Ack=553 Win=86 Len=0 TSval=9809642 TSecr=9809642
```

Practica 1B:

Analizar la salida en los terminales de los hosts (h2, h3 y h4) y estudiar el tráfico con Wireshark para explicar cómo se procesan los paquetes de petición HTTP. Se debe analizar especialmente la secuencia de mensajes Openflow y se deben adjuntar las capturas de pantalla necesarias.

Los paquetes se reparten a partes iguales entre h2, h3 y h4.

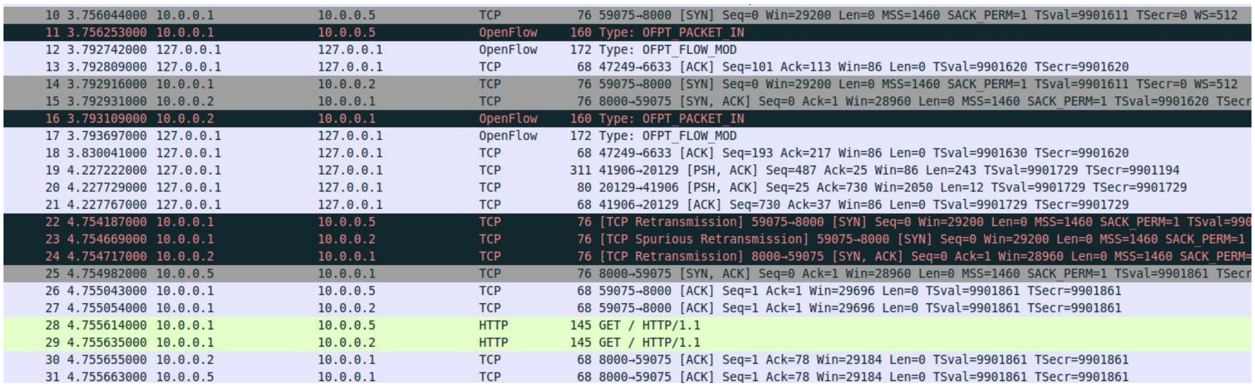


```
root@sdnhubvm:~/pox[16:24] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.1 - - [05/Nov/2022 16:25:58] "GET / HTTP/1.1" 200 -
```

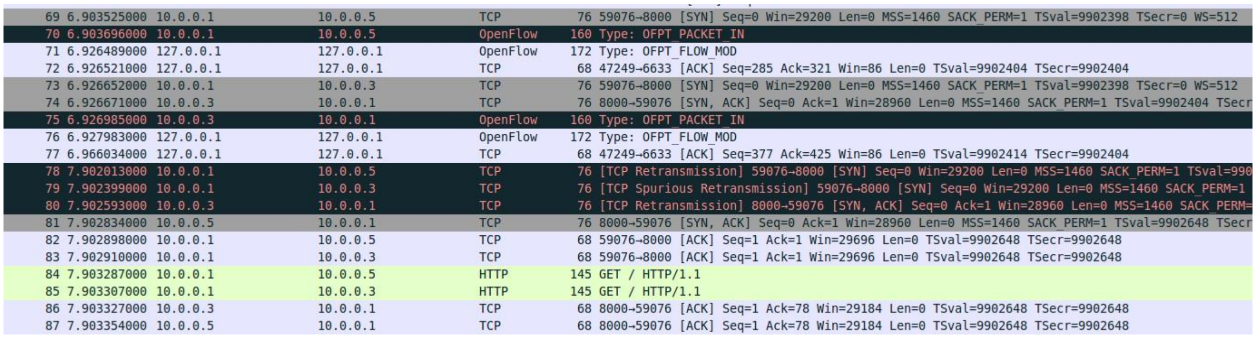
```
root@sdnhubvm:~/pox[16:24] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.1 - - [05/Nov/2022 16:26:01] "GET / HTTP/1.1" 200 -
```

```
root@sdnhubvm:~/pox[16:25] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.1 - - [05/Nov/2022 16:26:03] "GET / HTTP/1.1" 200 -
```

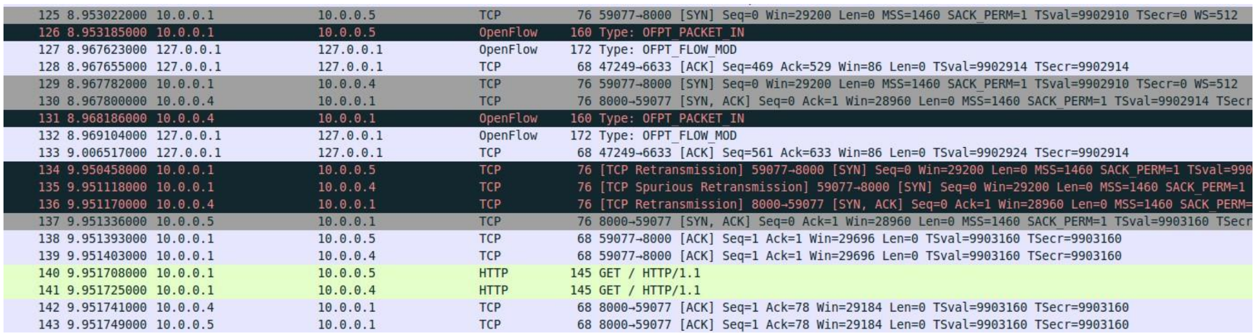
Los mensajes interesantes son los que están en gris. En la figura siguiente, se envía un primer paquete al equilibrador de carga (10.0.0.5), que no sabe qué hacer con él y lo envía al controlador. El controlador instala las reglas del equilibrador de carga (FLOW_MOD). El equilibrador de carga transfiere a h2 (10.0.0.2). h2 realiza la tarea solicitada y luego envía el paquete de vuelta al equilibrador de carga. Finalmente, el equilibrador de carga transfiere la respuesta de h2 a h1. Luego hace lo mismo con h3 para la siguiente solicitud y así sucesivamente.



```
   10 3.756044000 10.0.0.1              10.0.0.5              TCP        76 59075→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9901611 TSecr=0 WS=512
   11 3.756253000 10.0.0.1              10.0.0.5              OpenFlow  160 Type: OFPT_PACKET_IN
   12 3.792742000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
   13 3.792809000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=101 Ack=113 Win=86 Len=0 TSval=9901620 TSecr=9901620
   14 3.792916000 10.0.0.1              10.0.0.2              TCP        76 59075→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9901611 TSecr=0 WS=512
   15 3.792931000 10.0.0.2              10.0.0.1              TCP        76 8000→59075 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9901620 TSecr
   16 3.793109000 10.0.0.2              10.0.0.1              OpenFlow  160 Type: OFPT_PACKET_IN
   17 3.793697000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
   18 3.830041000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=193 Ack=217 Win=86 Len=0 TSval=9901630 TSecr=9901620
   19 4.227222000 127.0.0.1             127.0.0.1             TCP       311 41906→20129 [PSH, ACK] Seq=487 Ack=25 Win=86 Len=243 TSval=9901729 TSecr=9901194
   20 4.227729000 127.0.0.1             127.0.0.1             TCP        80 20129→41906 [PSH, ACK] Seq=25 Ack=730 Win=2050 Len=12 TSval=9901729 TSecr=9901729
   21 4.227767000 127.0.0.1             127.0.0.1             TCP        68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=9901729 TSecr=9901729
   22 4.754187000 10.0.0.1              10.0.0.5              TCP        76 [TCP Retransmission] 59075→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=990
   23 4.754669000 10.0.0.1              10.0.0.2              TCP        76 [TCP Spurious Retransmission] 59075→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
   24 4.754717000 10.0.0.2              10.0.0.1              TCP        76 [TCP Retransmission] 8000→59075 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
   25 4.754982000 10.0.0.5              10.0.0.1              TCP        76 8000→59075 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9901861 TSecr
   26 4.755043000 10.0.0.1              10.0.0.5              TCP        68 59075→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9901861 TSecr=9901861
   27 4.755054000 10.0.0.1              10.0.0.2              TCP        68 59075→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9901861 TSecr=9901861
   28 4.755614000 10.0.0.1              10.0.0.5              HTTP      145 GET / HTTP/1.1
   29 4.755635000 10.0.0.1              10.0.0.2              HTTP      145 GET / HTTP/1.1
   30 4.755655000 10.0.0.2              10.0.0.1              TCP        68 8000→59075 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9901861 TSecr=9901861
   31 4.755663000 10.0.0.5              10.0.0.1              TCP        68 8000→59075 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9901861 TSecr=9901861
```



```
   69 6.903525000 10.0.0.1              10.0.0.5              TCP        76 59076→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9902398 TSecr=0 WS=512
   70 6.903696000 10.0.0.1              10.0.0.5              OpenFlow  160 Type: OFPT_PACKET_IN
   71 6.926489000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
   72 6.926521000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=285 Ack=321 Win=86 Len=0 TSval=9902404 TSecr=9902404
   73 6.926652000 10.0.0.1              10.0.0.3              TCP        76 59076→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9902398 TSecr=0 WS=512
   74 6.926671000 10.0.0.3              10.0.0.1              TCP        76 8000→59076 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9902404 TSecr
   75 6.926985000 10.0.0.3              10.0.0.1              OpenFlow  160 Type: OFPT_PACKET_IN
   76 6.927983000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
   77 6.966034000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=377 Ack=425 Win=86 Len=0 TSval=9902414 TSecr=9902404
   78 7.902013000 10.0.0.1              10.0.0.5              TCP        76 [TCP Retransmission] 59076→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=990
   79 7.902399000 10.0.0.1              10.0.0.3              TCP        76 [TCP Spurious Retransmission] 59076→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
   80 7.902593000 10.0.0.3              10.0.0.1              TCP        76 [TCP Retransmission] 8000→59076 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
   81 7.902834000 10.0.0.5              10.0.0.1              TCP        76 8000→59076 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9902648 TSecr
   82 7.902898000 10.0.0.1              10.0.0.5              TCP        68 59076→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9902648 TSecr=9902648
   83 7.902910000 10.0.0.1              10.0.0.3              TCP        68 59076→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9902648 TSecr=9902648
   84 7.903287000 10.0.0.1              10.0.0.5              HTTP      145 GET / HTTP/1.1
   85 7.903307000 10.0.0.1              10.0.0.3              HTTP      145 GET / HTTP/1.1
   86 7.903327000 10.0.0.3              10.0.0.1              TCP        68 8000→59076 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9902648 TSecr=9902648
   87 7.903354000 10.0.0.5              10.0.0.1              TCP        68 8000→59076 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9902648 TSecr=9902648
```



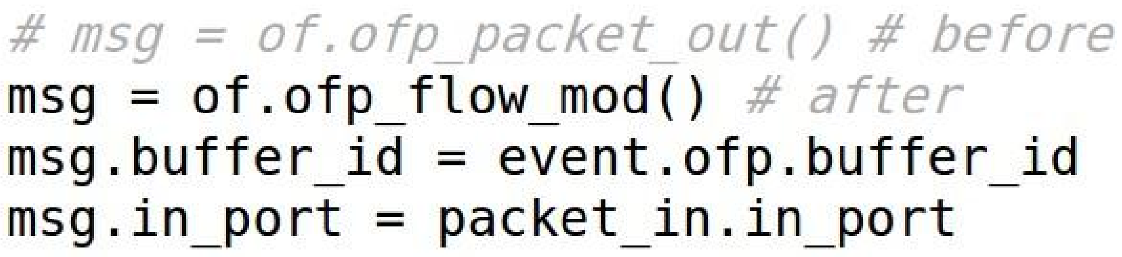
```
  125 8.953022000 10.0.0.1              10.0.0.5              TCP        76 59077→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9902910 TSecr=0 WS=512
  126 8.953185000 10.0.0.1              10.0.0.5              OpenFlow  160 Type: OFPT_PACKET_IN
  127 8.967623000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
  128 8.967655000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=469 Ack=529 Win=86 Len=0 TSval=9902914 TSecr=9902914
  129 8.967782000 10.0.0.1              10.0.0.4              TCP        76 59077→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9902910 TSecr=0 WS=512
  130 8.967800000 10.0.0.4              10.0.0.1              TCP        76 8000→59077 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9902914 TSecr
  131 8.968186000 10.0.0.4              10.0.0.1              OpenFlow  160 Type: OFPT_PACKET_IN
  132 8.969104000 127.0.0.1             127.0.0.1             OpenFlow  172 Type: OFPT_FLOW_MOD
  133 9.006517000 127.0.0.1             127.0.0.1             TCP        68 47249→6633 [ACK] Seq=561 Ack=633 Win=86 Len=0 TSval=9902924 TSecr=9902914
  134 9.950458000 10.0.0.1              10.0.0.5              TCP        76 [TCP Retransmission] 59077→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=990
  135 9.951118000 10.0.0.1              10.0.0.4              TCP        76 [TCP Spurious Retransmission] 59077→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
  136 9.951170000 10.0.0.4              10.0.0.1              TCP        76 [TCP Retransmission] 8000→59077 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=
  137 9.951336000 10.0.0.5              10.0.0.1              TCP        76 8000→59077 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=9903160 TSecr
  138 9.951393000 10.0.0.1              10.0.0.5              TCP        68 59077→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9903160 TSecr=9903160
  139 9.951403000 10.0.0.1              10.0.0.4              TCP        68 59077→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=9903160 TSecr=9903160
  140 9.951708000 10.0.0.1              10.0.0.5              HTTP      145 GET / HTTP/1.1
  141 9.951725000 10.0.0.1              10.0.0.4              HTTP      145 GET / HTTP/1.1
  142 9.951741000 10.0.0.4              10.0.0.1              TCP        68 8000→59077 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9903160 TSecr=9903160
  143 9.951749000 10.0.0.5              10.0.0.1              TCP        68 8000→59077 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=9903160 TSecr=9903160
```

Practica 2:

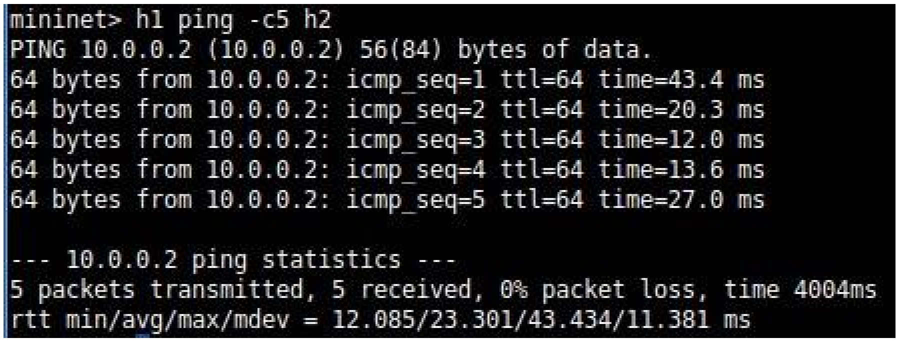
Crear reglas en el switch openflow (partiendo de tutorial_l2_hub) para ofrecer la misma funcionalidad que en el caso inicial, pero con un mejor rendimiento. En lugar de realizar la difusión desde la aplicación, lo que implica que cada trama ethernet sea procesada a nivel de controlador, se deben instalar reglas en el switch que eviten ese procesamiento. Emplear el comando ping para evaluar el Hub inicial y el optimizado para conocer el retraso que introduce la primera versión.

Cambiamos of.ofp_packet_out por of.ofp_flow_mod.

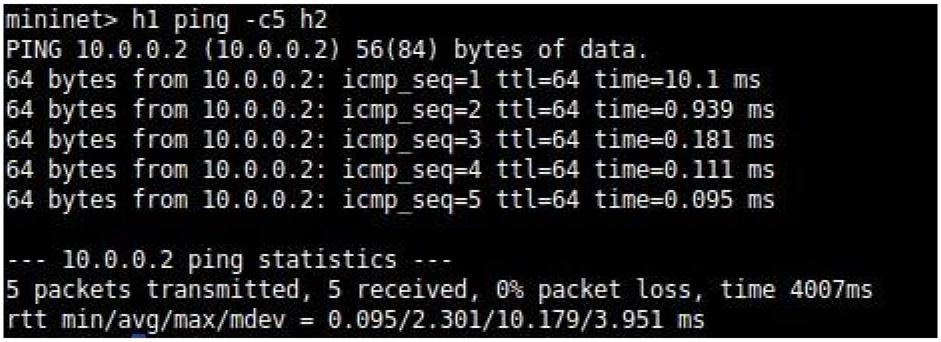
Esto cambia el tiempo medio de 23,301 ms a 2,301 ms.



```
# msg = of.ofp_packet_out() # before
msg = of.ofp_flow_mod() # after
msg.buffer_id = event.ofp.buffer_id
msg.in_port = packet_in.in_port
```



```
mininet> h1 ping -c5 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=43.4 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=12.0 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=13.6 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=27.0 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 12.085/23.301/43.434/11.381 ms
```

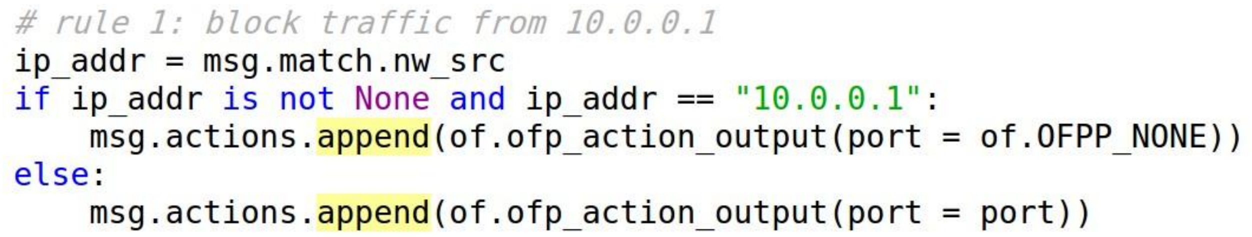


```
mininet> h1 ping -c5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=10.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.939 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.181 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.111 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.095 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 0.095/2.301/10.179/3.951 ms
```

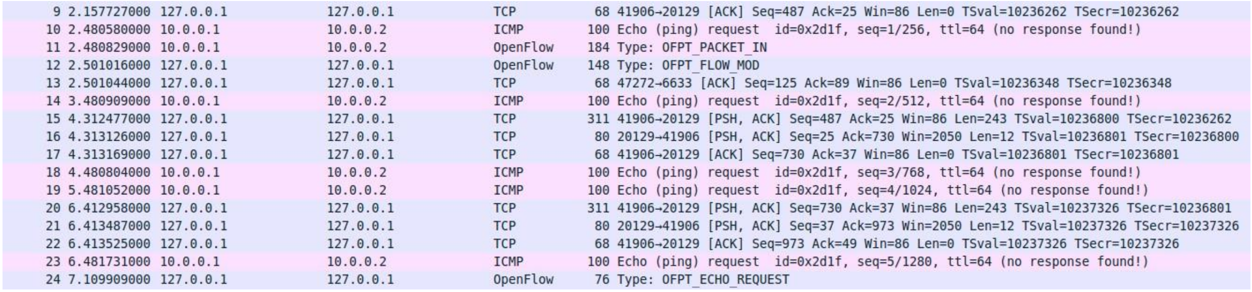
Crear un Firewall a partir de un switch estándar (partiendo de l2_learning) para que bloquee tráfico:

Ver archivo “l2_learning_copy.py”.

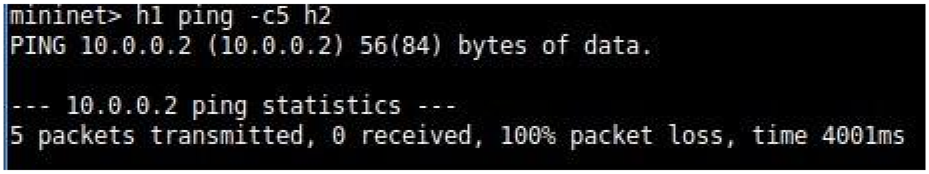
Desde un origen concreto a nivel IP.



```
# rule 1: block traffic from 10.0.0.1
ip_addr = msg.match.nw_src
if ip_addr is not None and ip_addr == "10.0.0.1":
    msg.actions.append(of.ofp_action_output(port = of.OFPP_NONE))
else:
    msg.actions.append(of.ofp_action_output(port = port))
```

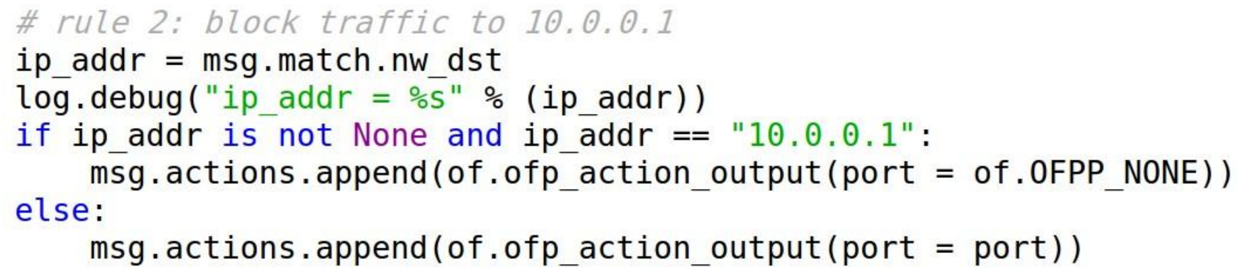


```
    9 2.157727000 127.0.0.1      127.0.0.1      TCP         68 41906→20129 [ACK] Seq=487 Ack=25 Win=86 Len=0 TSval=10236262 TSecr=10236262
   10 2.480580000 10.0.0.1       10.0.0.2       ICMP       100 Echo (ping) request  id=0x2d1f, seq=1/256, ttl=64 (no response found!)
   11 2.480829000 10.0.0.1       10.0.0.2       OpenFlow   184 Type: OFPT_PACKET_IN
   12 2.501016000 127.0.0.1      127.0.0.1      OpenFlow   148 Type: OFPT_FLOW_MOD
   13 2.501044000 127.0.0.1      127.0.0.1      TCP         68 47272→6633 [ACK] Seq=125 Ack=89 Win=86 Len=0 TSval=10236348 TSecr=10236348
   14 3.480909000 10.0.0.1       10.0.0.2       ICMP       100 Echo (ping) request  id=0x2d1f, seq=2/512, ttl=64 (no response found!)
   15 4.312477000 127.0.0.1      127.0.0.1      TCP        311 41906→20129 [PSH, ACK] Seq=487 Ack=25 Win=86 Len=243 TSval=10236800 TSecr=10236262
   16 4.313126000 127.0.0.1      127.0.0.1      TCP         80 20129→41906 [PSH, ACK] Seq=25 Ack=730 Win=2050 Len=12 TSval=10236801 TSecr=10236800
   17 4.313169000 127.0.0.1      127.0.0.1      TCP         68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=10236801 TSecr=10236801
   18 4.480804000 10.0.0.1       10.0.0.2       ICMP       100 Echo (ping) request  id=0x2d1f, seq=3/768, ttl=64 (no response found!)
   19 5.481052000 10.0.0.1       10.0.0.2       ICMP       100 Echo (ping) request  id=0x2d1f, seq=4/1024, ttl=64 (no response found!)
   20 6.412958000 127.0.0.1      127.0.0.1      TCP        311 41906→20129 [PSH, ACK] Seq=730 Ack=37 Win=86 Len=243 TSval=10237326 TSecr=10236801
   21 6.413487000 127.0.0.1      127.0.0.1      TCP         80 20129→41906 [PSH, ACK] Seq=37 Ack=973 Win=2050 Len=12 TSval=10237326 TSecr=10237326
   22 6.413525000 127.0.0.1      127.0.0.1      TCP         68 41906→20129 [ACK] Seq=973 Ack=49 Win=86 Len=0 TSval=10237326 TSecr=10237326
   23 6.481731000 10.0.0.1       10.0.0.2       ICMP       100 Echo (ping) request  id=0x2d1f, seq=5/1280, ttl=64 (no response found!)
   24 7.109909000 127.0.0.1      127.0.0.1      OpenFlow    76 Type: OFPT_ECHO_REQUEST
```

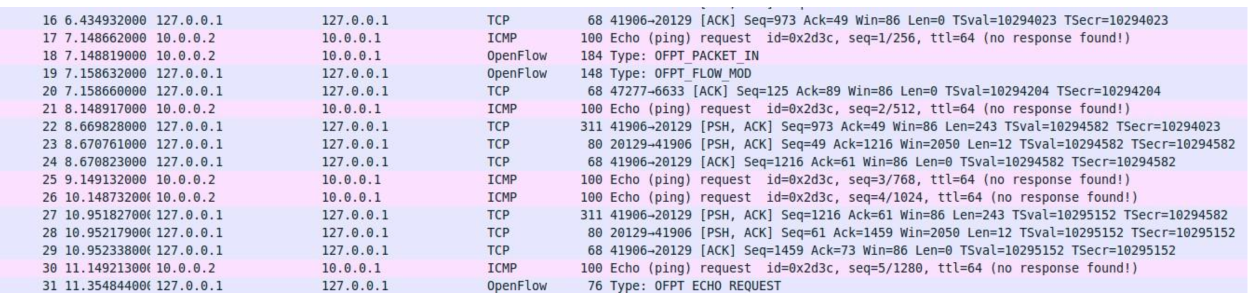


```
mininet> h1 ping -c5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4001ms
```

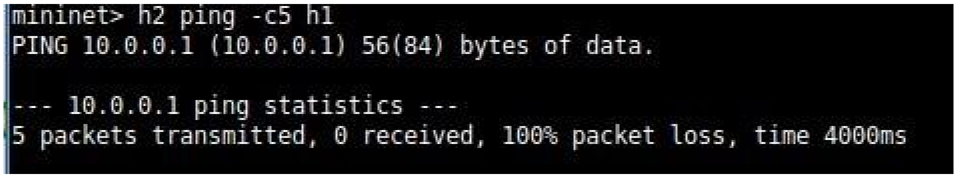
Con un destino concreto a nivel IP.



```
# rule 2: block traffic to 10.0.0.1
ip_addr = msg.match.nw_dst
log.debug("ip_addr = %s" % (ip_addr))
if ip_addr is not None and ip_addr == "10.0.0.1":
    msg.actions.append(of.ofp_action_output(port = of.OFPP_NONE))
else:
    msg.actions.append(of.ofp_action_output(port = port))
```



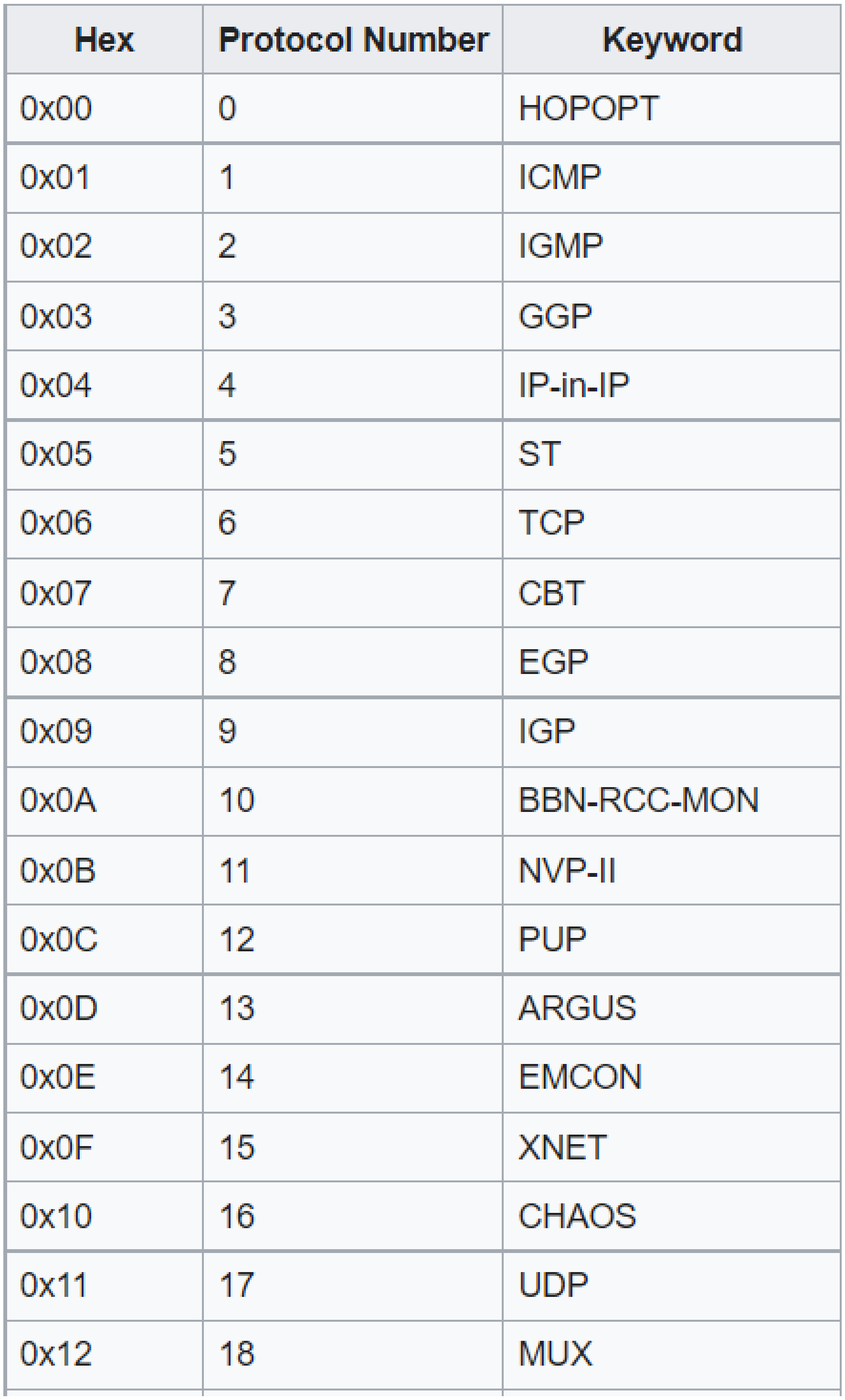
```
    16 6.434932000 127.0.0.1          127.0.0.1          TCP           68 41906→20129 [ACK] Seq=973 Ack=49 Win=86 Len=0 TSval=10294023 TSecr=10294023
    17 7.148662000 10.0.0.2           10.0.0.1           ICMP         100 Echo (ping) request  id=0x2d3c, seq=1/256, ttl=64 (no response found!)
    18 7.148819000 10.0.0.2           10.0.0.1           OpenFlow     184 Type: OFPT_PACKET_IN
    19 7.158632000 127.0.0.1          127.0.0.1          OpenFlow     148 Type: OFPT_FLOW_MOD
    20 7.158660000 127.0.0.1          127.0.0.1          TCP           68 47277→6633 [ACK] Seq=125 Ack=89 Win=86 Len=0 TSval=10294204 TSecr=10294204
    21 8.148917000 10.0.0.2           10.0.0.1           ICMP         100 Echo (ping) request  id=0x2d3c, seq=2/512, ttl=64 (no response found!)
    22 8.669828000 127.0.0.1          127.0.0.1          TCP          311 41906→20129 [PSH, ACK] Seq=973 Ack=49 Win=86 Len=243 TSval=10294582 TSecr=10294023
    23 8.670761000 127.0.0.1          127.0.0.1          TCP           80 20129→41906 [PSH, ACK] Seq=49 Ack=1216 Win=2050 Len=12 TSval=10294582 TSecr=10294582
    24 8.670823000 127.0.0.1          127.0.0.1          TCP           68 41906→20129 [ACK] Seq=1216 Ack=61 Win=86 Len=0 TSval=10294582 TSecr=10294582
    25 9.149132000 10.0.0.2           10.0.0.1           ICMP         100 Echo (ping) request  id=0x2d3c, seq=3/768, ttl=64 (no response found!)
    26 10.14873200( 10.0.0.2          10.0.0.1           ICMP         100 Echo (ping) request  id=0x2d3c, seq=4/1024, ttl=64 (no response found!)
    27 10.95182700( 127.0.0.1         127.0.0.1          TCP          311 41906→20129 [PSH, ACK] Seq=1216 Ack=61 Win=86 Len=243 TSval=10295152 TSecr=10294582
    28 10.95217900( 127.0.0.1         127.0.0.1          TCP           80 20129→41906 [PSH, ACK] Seq=61 Ack=1459 Win=2050 Len=12 TSval=10295152 TSecr=10295152
    29 10.95233800( 127.0.0.1         127.0.0.1          TCP           68 41906→20129 [ACK] Seq=1459 Ack=73 Win=86 Len=0 TSval=10295152 TSecr=10295152
    30 11.14921300( 10.0.0.2          10.0.0.1           ICMP         100 Echo (ping) request  id=0x2d3c, seq=5/1280, ttl=64 (no response found!)
    31 11.35484400( 127.0.0.1         127.0.0.1          OpenFlow      76 Type: OFPT_ECHO_REQUEST
```



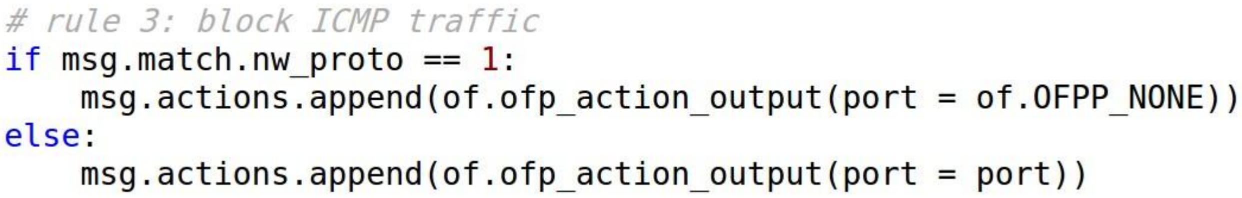
```
mininet> h2 ping -c5 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4000ms
```

Tráfico con el protocolo ICMP.

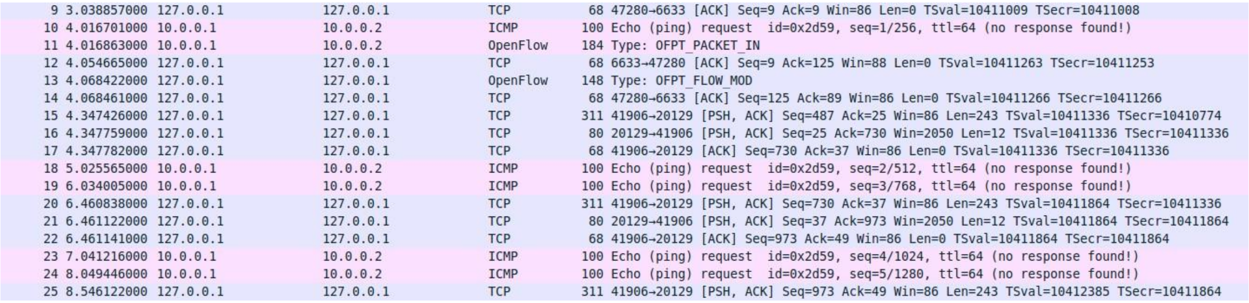
Para “match” los protocolos, he utilizado los números de protocolo IP.



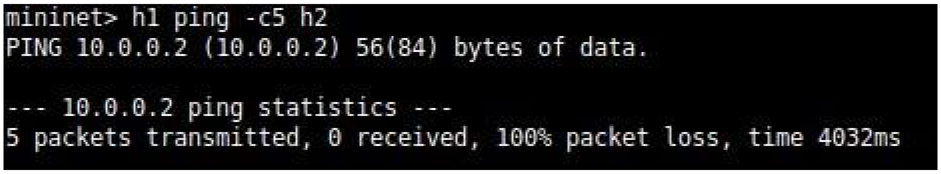
Hex	Protocol Number	Keyword
0x00	0	HOPOPT
0x01	1	ICMP
0x02	2	IGMP
0x03	3	GGP
0x04	4	IP-in-IP
0x05	5	ST
0x06	6	TCP
0x07	7	CBT
0x08	8	EGP
0x09	9	IGP
0x0A	10	BBN-RCC-MON
0x0B	11	NVP-II
0x0C	12	PUP
0x0D	13	ARGUS
0x0E	14	EMCON
0x0F	15	XNET
0x10	16	CHAOS
0x11	17	UDP
0x12	18	MUX



```
# rule 3: block ICMP traffic
if msg.match.nw_proto == 1:
    msg.actions.append(of.ofp_action_output(port = of.OFPP_NONE))
else:
    msg.actions.append(of.ofp_action_output(port = port))
```

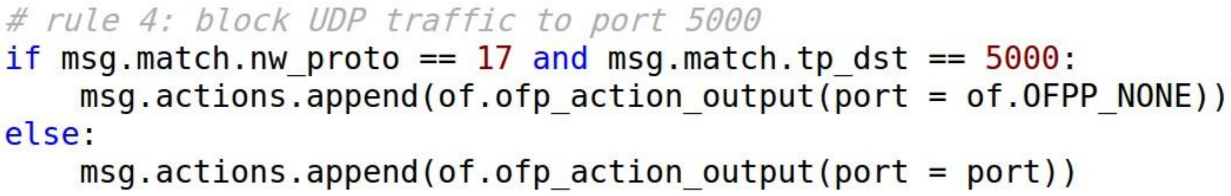


```
  9 3.038857000 127.0.0.1       127.0.0.1       TCP        68 47280→6633 [ACK] Seq=9 Ack=9 Win=86 Len=0 TSval=10411009 TSecr=10411008
 10 4.016701000 10.0.0.1        10.0.0.2        ICMP      100 Echo (ping) request  id=0x2d59, seq=1/256, ttl=64 (no response found!)
 11 4.016863000 10.0.0.1        10.0.0.2        OpenFlow  184 Type: OFPT_PACKET_IN
 12 4.054665000 127.0.0.1       127.0.0.1       TCP        68 6633→47280 [ACK] Seq=9 Ack=125 Win=88 Len=0 TSval=10411263 TSecr=10411253
 13 4.068422000 127.0.0.1       127.0.0.1       OpenFlow  148 Type: OFPT_FLOW_MOD
 14 4.068461000 127.0.0.1       127.0.0.1       TCP        68 47280→6633 [ACK] Seq=125 Ack=89 Win=86 Len=0 TSval=10411266 TSecr=10411266
 15 4.347426000 127.0.0.1       127.0.0.1       TCP       311 41906→20129 [PSH, ACK] Seq=487 Ack=25 Win=86 Len=243 TSval=10411336 TSecr=10410774
 16 4.347759000 127.0.0.1       127.0.0.1       TCP        80 20129→41906 [PSH, ACK] Seq=25 Ack=730 Win=2050 Len=12 TSval=10411336 TSecr=10411336
 17 4.347782000 127.0.0.1       127.0.0.1       TCP        68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=10411336 TSecr=10411336
 18 5.025565000 10.0.0.1        10.0.0.2        ICMP      100 Echo (ping) request  id=0x2d59, seq=2/512, ttl=64 (no response found!)
 19 6.034005000 10.0.0.1        10.0.0.2        ICMP      100 Echo (ping) request  id=0x2d59, seq=3/768, ttl=64 (no response found!)
 20 6.460838000 127.0.0.1       127.0.0.1       TCP       311 41906→20129 [PSH, ACK] Seq=730 Ack=37 Win=86 Len=243 TSval=10411864 TSecr=10411336
 21 6.461122000 127.0.0.1       127.0.0.1       TCP        80 20129→41906 [PSH, ACK] Seq=37 Ack=973 Win=2050 Len=12 TSval=10411864 TSecr=10411864
 22 6.461141000 127.0.0.1       127.0.0.1       TCP        68 41906→20129 [ACK] Seq=973 Ack=49 Win=86 Len=0 TSval=10411864 TSecr=10411864
 23 7.041216000 10.0.0.1        10.0.0.2        ICMP      100 Echo (ping) request  id=0x2d59, seq=4/1024, ttl=64 (no response found!)
 24 8.049446000 10.0.0.1        10.0.0.2        ICMP      100 Echo (ping) request  id=0x2d59, seq=5/1280, ttl=64 (no response found!)
 25 8.546122000 127.0.0.1       127.0.0.1       TCP       311 41906→20129 [PSH, ACK] Seq=973 Ack=49 Win=86 Len=243 TSval=10412385 TSecr=10411864
```

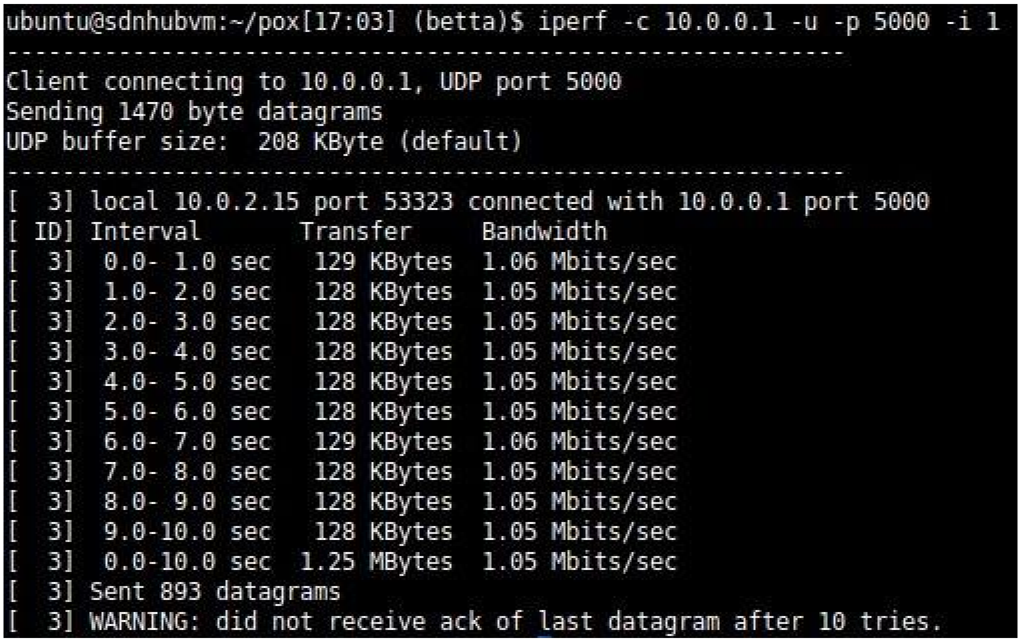


```
mininet> h1 ping -c5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4032ms
```

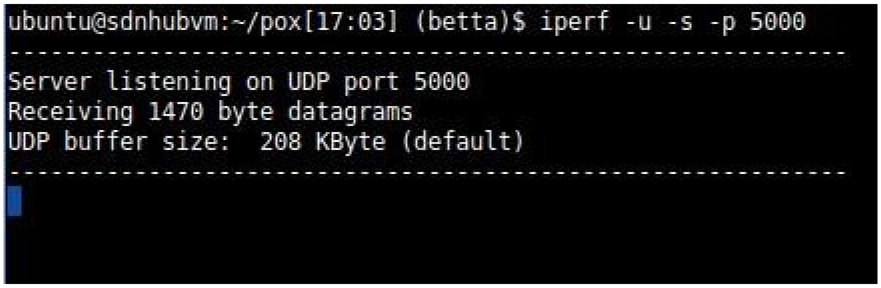
Tráfico UDP a un puerto concreto.



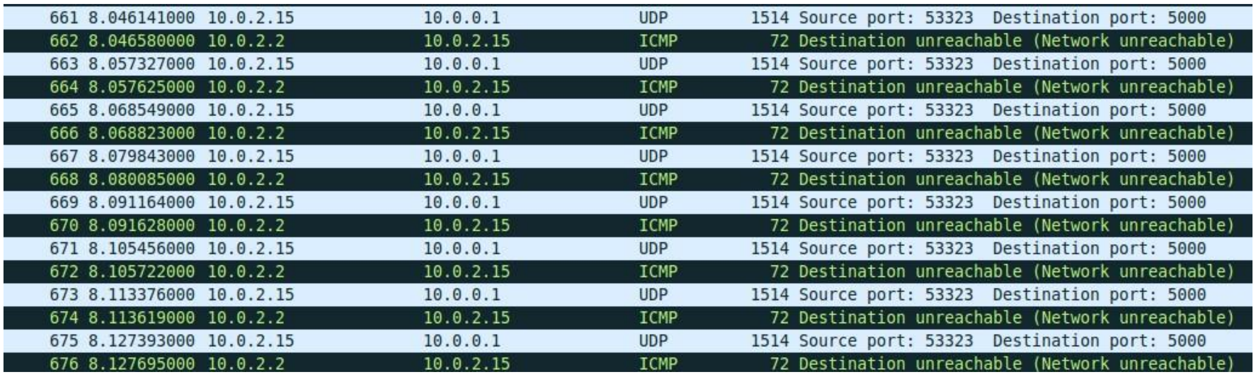
```
# rule 4: block UDP traffic to port 5000
if msg.match.nw_proto == 17 and msg.match.tp_dst == 5000:
    msg.actions.append(of.ofp_action_output(port = of.OFPP_NONE))
else:
    msg.actions.append(of.ofp_action_output(port = port))
```



```
ubuntu@sdnhubvm:~/pox[17:03] (betta)$ iperf -c 10.0.0.1 -u -p 5000 -i 1
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5000
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.2.15 port 53323 connected with 10.0.0.1 port 5000
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
```

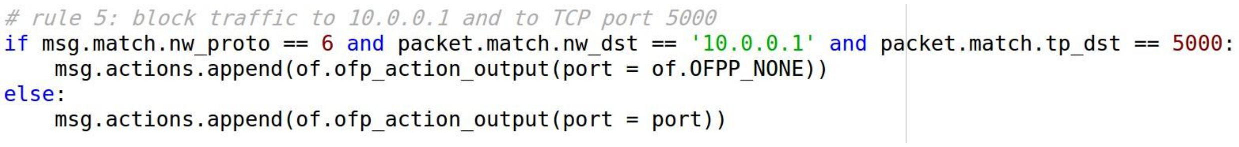


```
ubuntu@sdnhubvm:~/pox[17:03] (betta)$ iperf -u -s -p 5000
------------------------------------------------------------
Server listening on UDP port 5000
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

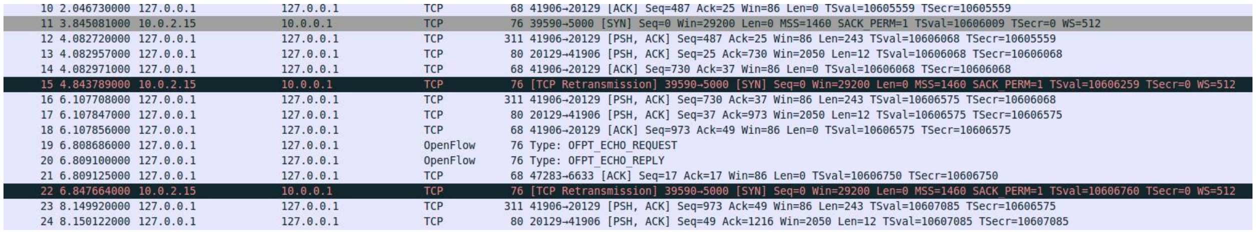


```
661 8.046141000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
662 8.046580000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
663 8.057327000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
664 8.057625000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
665 8.068549000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
666 8.068823000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
667 8.079843000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
668 8.080085000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
669 8.091164000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
670 8.091628000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
671 8.105456000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
672 8.105722000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
673 8.113376000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
674 8.113619000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
675 8.127393000  10.0.2.15      10.0.0.1       UDP     1514 Source port: 53323  Destination port: 5000
676 8.127695000  10.0.2.2       10.0.2.15      ICMP      72 Destination unreachable (Network unreachable)
```

Tráfico con un destino a nivel IP y puerto TCP concretos.



```
# rule 5: block traffic to 10.0.0.1 and to TCP port 5000
if msg.match.nw_proto == 6 and packet.match.nw_dst == '10.0.0.1' and packet.match.tp_dst == 5000:
    msg.actions.append(of.ofp_action_output(port = of.OFPP_NONE))
else:
    msg.actions.append(of.ofp_action_output(port = port))
```



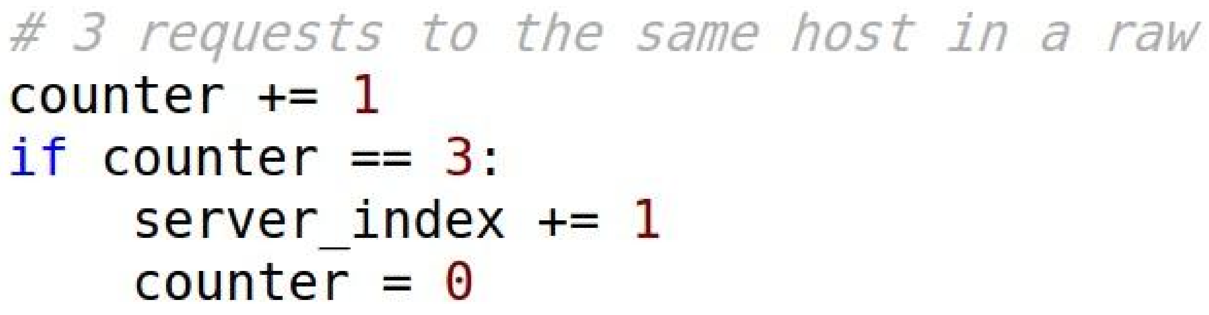
```
10 2.046730000  127.0.0.1   127.0.0.1   TCP        68 41906→20129 [ACK] Seq=487 Ack=25 Win=86 Len=0 TSval=10605559 TSecr=10605559
11 3.845081000  10.0.2.15   10.0.0.1    TCP        76 39590→5000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10606009 TSecr=0 WS=512
12 4.082720000  127.0.0.1   127.0.0.1   TCP       311 41906→20129 [PSH, ACK] Seq=487 Ack=25 Win=86 Len=243 TSval=10606068 TSecr=10605559
13 4.082957000  127.0.0.1   127.0.0.1   TCP        80 20129→41906 [PSH, ACK] Seq=25 Ack=730 Win=2050 Len=12 TSval=10606068 TSecr=10606068
14 4.082971000  127.0.0.1   127.0.0.1   TCP        68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=10606068 TSecr=10606068
15 4.843789000  10.0.2.15   10.0.0.1    TCP        76 [TCP Retransmission] 39590→5000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10606259 TSecr=0 WS=512
16 6.107708000  127.0.0.1   127.0.0.1   TCP       311 41906→20129 [PSH, ACK] Seq=730 Ack=37 Win=86 Len=243 TSval=10606575 TSecr=10606068
17 6.107847000  127.0.0.1   127.0.0.1   TCP        80 20129→41906 [PSH, ACK] Seq=37 Ack=973 Win=2050 Len=12 TSval=10606575 TSecr=10606575
18 6.107856000  127.0.0.1   127.0.0.1   TCP        68 41906→20129 [ACK] Seq=973 Ack=49 Win=86 Len=0 TSval=10606575 TSecr=10606575
19 6.808686000  127.0.0.1   127.0.0.1   OpenFlow   76 Type: OFPT_ECHO_REQUEST
20 6.809100000  127.0.0.1   127.0.0.1   OpenFlow   76 Type: OFPT_ECHO_REPLY
21 6.809125000  127.0.0.1   127.0.0.1   TCP        68 47283→6633 [ACK] Seq=17 Ack=17 Win=86 Len=0 TSval=10606750 TSecr=10606750
22 6.847664000  10.0.2.15   10.0.0.1    TCP        76 [TCP Retransmission] 39590→5000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10606760 TSecr=0 WS=512
23 8.149920000  127.0.0.1   127.0.0.1   TCP       311 41906→20129 [PSH, ACK] Seq=973 Ack=49 Win=86 Len=243 TSval=10607085 TSecr=10606575
24 8.150122000  127.0.0.1   127.0.0.1   TCP        80 20129→41906 [PSH, ACK] Seq=49 Ack=1216 Win=2050 Len=12 TSval=10607085 TSecr=10607085
```

Practica 3:

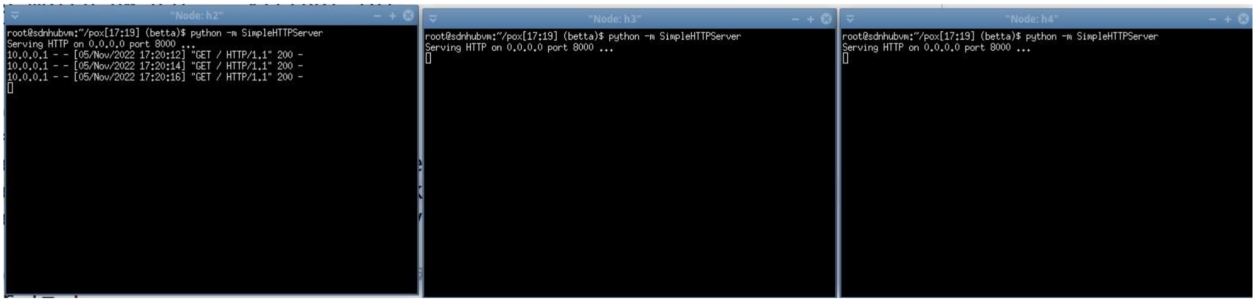
Modificar el balanceador (aplicación tutorial_stateless_lb) para que:

Se envíen 3 peticiones seguidas hacia el mismo servidor.

Ver archivo “tutorial_stateless_lb_copy.py”.



```
# 3 requests to the same host in a raw
counter += 1
if counter == 3:
        server_index += 1
        counter = 0
```



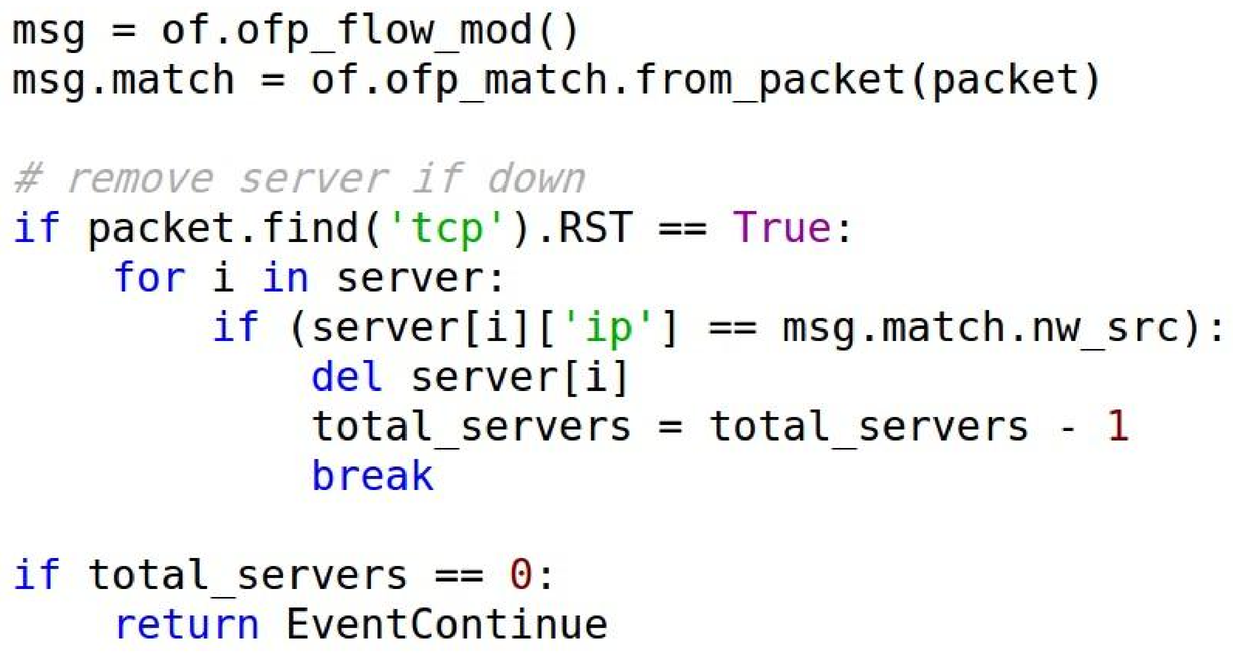
```
root@sdnhubvm:~/pox[17:19] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.1 - - [05/Nov/2022 17:20:12] "GET / HTTP/1.1" 200 -
10.0.0.1 - - [05/Nov/2022 17:20:14] "GET / HTTP/1.1" 200 -
10.0.0.1 - - [05/Nov/2022 17:20:16] "GET / HTTP/1.1" 200 -
```

```
root@sdnhubvm:~/pox[17:19] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

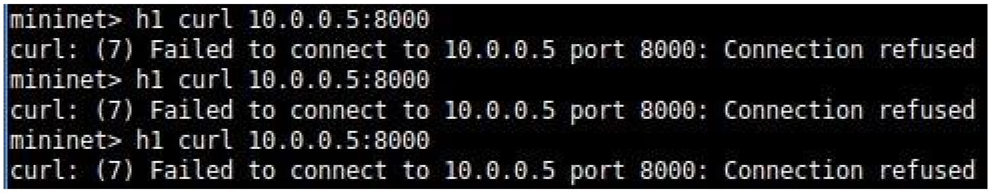
```
root@sdnhubvm:~/pox[17:19] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Detecte si un servidor web ha caído (aunque el ordenador sigue conectado a la red) y, en ese caso, eliminarlo de la lista de servidores del pool.

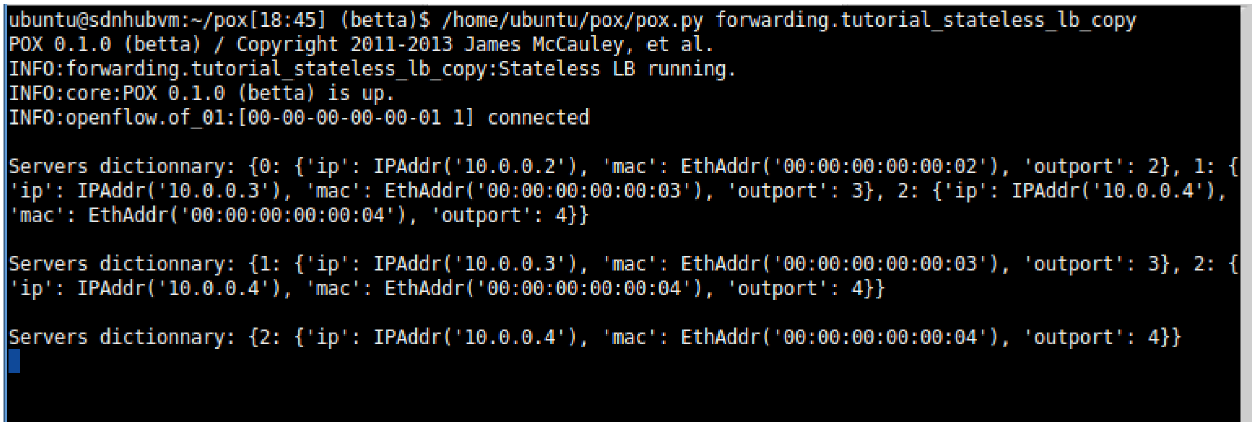
Ver archivo “tutorial_stateless_lb_copy.py”. Lo importante es que cuando un servidor está caído y no puede ser alcanzado, se recibe un ACK RST.



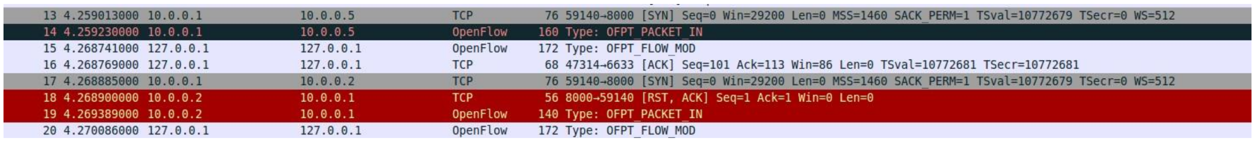
```
msg = of.ofp_flow_mod()
msg.match = of.ofp_match.from_packet(packet)

# remove server if down
if packet.find('tcp').RST == True:
    for i in server:
        if (server[i]['ip'] == msg.match.nw_src):
            del server[i]
            total_servers = total_servers - 1
            break

if total_servers == 0:
    return EventContinue
```



```
mininet> h1 curl 10.0.0.5:8000
curl: (7) Failed to connect to 10.0.0.5 port 8000: Connection refused
mininet> h1 curl 10.0.0.5:8000
curl: (7) Failed to connect to 10.0.0.5 port 8000: Connection refused
mininet> h1 curl 10.0.0.5:8000
curl: (7) Failed to connect to 10.0.0.5 port 8000: Connection refused
```



```
ubuntu@sdnhubvm:~/pox[18:45] (betta)$ /home/ubuntu/pox/pox.py forwarding.tutorial_stateless_lb_copy
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.tutorial_stateless_lb_copy:Stateless LB running.
INFO:core:POX 0.1.0 (betta) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected

Servers dictionnary: {0: {'ip': IPAddr('10.0.0.2'), 'mac': EthAddr('00:00:00:00:00:02'), 'outport': 2}, 1: {
'ip': IPAddr('10.0.0.3'), 'mac': EthAddr('00:00:00:00:00:03'), 'outport': 3}, 2: {'ip': IPAddr('10.0.0.4'),
'mac': EthAddr('00:00:00:00:00:04'), 'outport': 4}}

Servers dictionnary: {1: {'ip': IPAddr('10.0.0.3'), 'mac': EthAddr('00:00:00:00:00:03'), 'outport': 3}, 2: {
'ip': IPAddr('10.0.0.4'), 'mac': EthAddr('00:00:00:00:00:04'), 'outport': 4}}

Servers dictionnary: {2: {'ip': IPAddr('10.0.0.4'), 'mac': EthAddr('00:00:00:00:00:04'), 'outport': 4}}
```

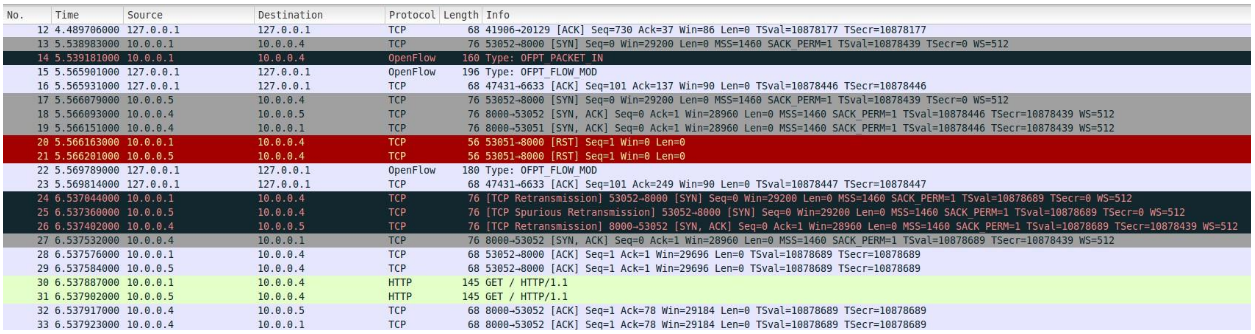


```
13 4.259013000 10.0.0.1     10.0.0.5     TCP       76 59140→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10772679 TSecr=0 WS=512
14 4.259230000 10.0.0.1     10.0.0.5     OpenFlow 160 Type: OFPT_PACKET_IN
15 4.268741000 127.0.0.1    127.0.0.1    OpenFlow 172 Type: OFPT_FLOW_MOD
16 4.268769000 127.0.0.1    127.0.0.1    TCP       68 47314→6633 [ACK] Seq=101 Ack=113 Win=86 Len=0 TSval=10772681 TSecr=10772681
17 4.268885000 10.0.0.1     10.0.0.2     TCP       76 59140→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10772679 TSecr=0 WS=512
18 4.268900000 10.0.0.2     10.0.0.1     TCP       56 8000→59140 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19 4.269389000 10.0.0.2     10.0.0.1     OpenFlow 140 Type: OFPT_PACKET_IN
20 4.270086000 127.0.0.1    127.0.0.1    OpenFlow 172 Type: OFPT_FLOW_MOD
```

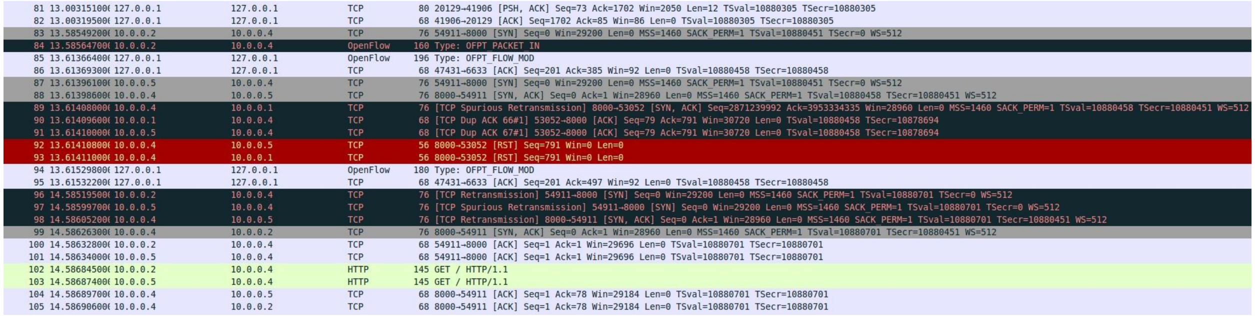
Inspirarse en el balanceador de carga para crear una aplicación que implemente un traductor de direcciones (NAT).

Ver archivo “NAT.py”. Funciona bien. Sólo he utilizado un servidor web (10.0.0.4), el router virtual (10.0.0.5) y tres hosts (10.0.0.1/2/3).

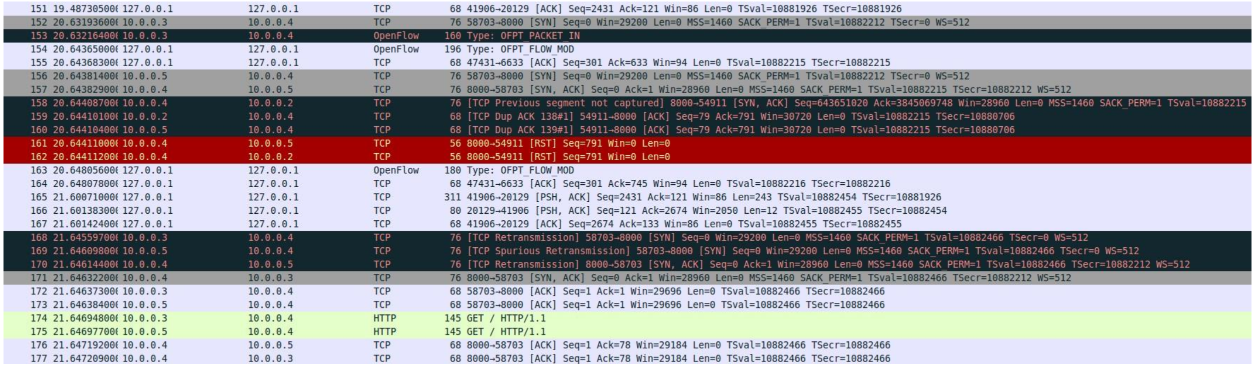
Ejecuto *h1 curl 10.0.0.4:8000*, luego *h2 curl 10.0.0.4:8000*, y finalmente *h3 curl 10.0.0.4:8000.*



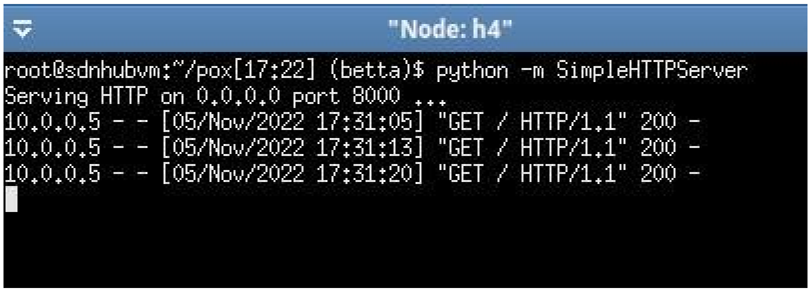
```
No.   Time        Source      Destination  Protocol Length Info
   12 4.489706000 127.0.0.1   127.0.0.1    TCP        68 41906→20129 [ACK] Seq=730 Ack=37 Win=86 Len=0 TSval=10878177 TSecr=10878177
   13 5.538983000 10.0.0.1    10.0.0.4     TCP        76 53052→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10878439 TSecr=0 WS=512
   14 5.539181000 10.0.0.1    10.0.0.4     OpenFlow  160 Type: OFPT_PACKET_IN
   15 5.565901000 127.0.0.1   127.0.0.1    OpenFlow  196 Type: OFPT_FLOW_MOD
   16 5.565931000 127.0.0.1   127.0.0.1    TCP        68 47431→6633 [ACK] Seq=101 Ack=137 Win=90 Len=0 TSval=10878446 TSecr=10878446
   17 5.566079000 10.0.0.5    10.0.0.4     TCP        76 53052→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10878439 TSecr=0 WS=512
   18 5.566093000 10.0.0.4    10.0.0.5     TCP        76 8000→53052 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10878446 TSecr=10878439 WS=512
   19 5.566151000 10.0.0.4    10.0.0.1     TCP        76 8000→53051 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10878446 TSecr=10878439 WS=512
   20 5.566163000 10.0.0.1    10.0.0.4     TCP        56 53051→8000 [RST] Seq=1 Win=0 Len=0
   21 5.566201000 10.0.0.5    10.0.0.4     TCP        56 53051→8000 [RST] Seq=1 Win=0 Len=0
   22 5.569789000 127.0.0.1   127.0.0.1    OpenFlow  180 Type: OFPT_FLOW_MOD
   23 5.569814000 127.0.0.1   127.0.0.1    TCP        68 47431→6633 [ACK] Seq=101 Ack=249 Win=90 Len=0 TSval=10878447 TSecr=10878447
   24 6.537044000 10.0.0.1    10.0.0.4     TCP        76 [TCP Retransmission] 53052→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10878689 TSecr=0 WS=512
   25 6.537360000 10.0.0.5    10.0.0.4     TCP        76 [TCP Spurious Retransmission] 53052→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10878689 TSecr=0 WS=512
   26 6.537402000 10.0.0.4    10.0.0.5     TCP        76 [TCP Retransmission] 8000→53052 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10878689 TSecr=10878439 WS=512
   27 6.537532000 10.0.0.4    10.0.0.1     TCP        76 8000→53052 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10878689 TSecr=10878439 WS=512
   28 6.537576000 10.0.0.1    10.0.0.4     TCP        68 53052→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10878689 TSecr=10878689
   29 6.537584000 10.0.0.5    10.0.0.4     TCP        68 53052→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10878689 TSecr=10878689
   30 6.537887000 10.0.0.1    10.0.0.4     HTTP      145 GET / HTTP/1.1
   31 6.537902000 10.0.0.5    10.0.0.4     HTTP      145 GET / HTTP/1.1
   32 6.537917000 10.0.0.4    10.0.0.5     TCP        68 8000→53052 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10878689 TSecr=10878689
   33 6.537923000 10.0.0.4    10.0.0.1     TCP        68 8000→53052 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10878689 TSecr=10878689
```



```
   81 13.00315100( 127.0.0.1  127.0.0.1   TCP        80 20129→41906 [PSH, ACK] Seq=73 Ack=1702 Win=2050 Len=12 TSval=10880305 TSecr=10880305
   82 13.00319500( 127.0.0.1  127.0.0.1   TCP        68 41906→20129 [ACK] Seq=1702 Ack=85 Win=86 Len=0 TSval=10880305 TSecr=10880305
   83 13.58549200( 10.0.0.2   10.0.0.4    TCP        76 54911→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10880451 TSecr=0 WS=512
   84 13.58564700( 10.0.0.2   10.0.0.4    OpenFlow  160 Type: OFPT_PACKET_IN
   85 13.61366400( 127.0.0.1  127.0.0.1   OpenFlow  196 Type: OFPT_FLOW_MOD
   86 13.61369300( 127.0.0.1  127.0.0.1   TCP        68 47431→6633 [ACK] Seq=201 Ack=385 Win=92 Len=0 TSval=10880458 TSecr=10880458
   87 13.61396100( 10.0.0.5   10.0.0.4    TCP        76 54911→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10880451 TSecr=0 WS=512
   88 13.61398600( 10.0.0.4   10.0.0.5    TCP        76 8000→54911 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10880458 TSecr=10880451 WS=512
   89 13.61408000( 10.0.0.4   10.0.0.1    TCP        76 [TCP Spurious Retransmission] 8000→53052 [SYN, ACK] Seq=2871239992 Ack=3953334335 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10880458 TSecr=10880451 WS=512
   90 13.61409600( 10.0.0.1   10.0.0.4    TCP        68 [TCP Dup ACK 66#1] 53052→8000 [ACK] Seq=79 Ack=791 Win=30720 Len=0 TSval=10880458 TSecr=10878694
   91 13.61410000( 10.0.0.5   10.0.0.4    TCP        68 [TCP Dup ACK 67#1] 53052→8000 [ACK] Seq=79 Ack=791 Win=30720 Len=0 TSval=10880458 TSecr=10878694
   92 13.61410800( 10.0.0.4   10.0.0.5    TCP        56 8000→53052 [RST] Seq=791 Win=0 Len=0
   93 13.61411000( 10.0.0.4   10.0.0.1    TCP        56 8000→53052 [RST] Seq=791 Win=0 Len=0
   94 13.61529800( 127.0.0.1  127.0.0.1   OpenFlow  180 Type: OFPT_FLOW_MOD
   95 13.61532200( 127.0.0.1  127.0.0.1   TCP        68 47431→6633 [ACK] Seq=201 Ack=497 Win=92 Len=0 TSval=10880458 TSecr=10880458
   96 14.58519500( 10.0.0.2   10.0.0.4    TCP        76 [TCP Retransmission] 54911→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10880701 TSecr=0 WS=512
   97 14.58599700( 10.0.0.5   10.0.0.4    TCP        76 [TCP Spurious Retransmission] 54911→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10880701 TSecr=0 WS=512
   98 14.58605200( 10.0.0.4   10.0.0.5    TCP        76 [TCP Retransmission] 8000→54911 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10880701 TSecr=10880451 WS=512
   99 14.58626300( 10.0.0.4   10.0.0.2    TCP        76 8000→54911 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10880701 TSecr=10880451 WS=512
  100 14.58632800( 10.0.0.2   10.0.0.4    TCP        68 54911→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10880701 TSecr=10880701
  101 14.58634000( 10.0.0.5   10.0.0.4    TCP        68 54911→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10880701 TSecr=10880701
  102 14.58684500( 10.0.0.2   10.0.0.4    HTTP      145 GET / HTTP/1.1
  103 14.58687400( 10.0.0.5   10.0.0.4    HTTP      145 GET / HTTP/1.1
  104 14.58689700( 10.0.0.4   10.0.0.5    TCP        68 8000→54911 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10880701 TSecr=10880701
  105 14.58690600( 10.0.0.4   10.0.0.2    TCP        68 8000→54911 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10880701 TSecr=10880701
```



```
  151 19.48730500( 127.0.0.1  127.0.0.1   TCP        68 41906→20129 [ACK] Seq=2431 Ack=121 Win=86 Len=0 TSval=10881926 TSecr=10881926
  152 20.63193600( 10.0.0.3   10.0.0.4    TCP        76 58703→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10882212 TSecr=0 WS=512
  153 20.63216400( 10.0.0.3   10.0.0.4    OpenFlow  160 Type: OFPT_PACKET_IN
  154 20.64365000( 127.0.0.1  127.0.0.1   OpenFlow  196 Type: OFPT_FLOW_MOD
  155 20.64368300( 127.0.0.1  127.0.0.1   TCP        68 47431→6633 [ACK] Seq=301 Ack=633 Win=94 Len=0 TSval=10882215 TSecr=10882215
  156 20.64381400( 10.0.0.5   10.0.0.4    TCP        76 58703→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10882212 TSecr=0 WS=512
  157 20.64382900( 10.0.0.4   10.0.0.5    TCP        76 8000→58703 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10882215 TSecr=10882212 WS=512
  158 20.64408700( 10.0.0.4   10.0.0.2    TCP        76 [TCP Previous segment not captured] 8000→54911 [SYN, ACK] Seq=643651020 Ack=3845069748 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10882215
  159 20.64410100( 10.0.0.2   10.0.0.4    TCP        68 [TCP Dup ACK 138#1] 54911→8000 [ACK] Seq=79 Ack=791 Win=30720 Len=0 TSval=10882215 TSecr=10880706
  160 20.64410400( 10.0.0.5   10.0.0.4    TCP        68 [TCP Dup ACK 139#1] 54911→8000 [ACK] Seq=79 Ack=791 Win=30720 Len=0 TSval=10882215 TSecr=10880706
  161 20.64411000( 10.0.0.4   10.0.0.5    TCP        56 8000→54911 [RST] Seq=791 Win=0 Len=0
  162 20.64411200( 10.0.0.4   10.0.0.2    TCP        56 8000→54911 [RST] Seq=791 Win=0 Len=0
  163 20.64805600( 127.0.0.1  127.0.0.1   OpenFlow  180 Type: OFPT_FLOW_MOD
  164 20.64807800( 127.0.0.1  127.0.0.1   TCP        68 47431→6633 [ACK] Seq=301 Ack=745 Win=94 Len=0 TSval=10882216 TSecr=10882216
  165 21.60071000( 127.0.0.1  127.0.0.1   TCP       311 41906→20129 [PSH, ACK] Seq=2431 Ack=121 Win=86 Len=243 TSval=10882454 TSecr=10881926
  166 21.60138300( 127.0.0.1  127.0.0.1   TCP        80 20129→41906 [PSH, ACK] Seq=121 Ack=2674 Win=2050 Len=12 TSval=10882455 TSecr=10882454
  167 21.60142400( 127.0.0.1  127.0.0.1   TCP        68 41906→20129 [ACK] Seq=2674 Ack=133 Win=86 Len=0 TSval=10882455 TSecr=10882455
  168 21.64559700( 10.0.0.3   10.0.0.4    TCP        76 [TCP Retransmission] 58703→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10882466 TSecr=0 WS=512
  169 21.64609800( 10.0.0.5   10.0.0.4    TCP        76 [TCP Spurious Retransmission] 58703→8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10882466 TSecr=0 WS=512
  170 21.64614400( 10.0.0.4   10.0.0.5    TCP        76 [TCP Retransmission] 8000→58703 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10882466 TSecr=10882212 WS=512
  171 21.64632200( 10.0.0.4   10.0.0.3    TCP        76 8000→58703 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10882466 TSecr=10882212 WS=512
  172 21.64637300( 10.0.0.3   10.0.0.4    TCP        68 58703→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10882466 TSecr=10882466
  173 21.64638400( 10.0.0.5   10.0.0.4    TCP        68 58703→8000 [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=10882466 TSecr=10882466
  174 21.64694800( 10.0.0.3   10.0.0.4    HTTP      145 GET / HTTP/1.1
  175 21.64697700( 10.0.0.5   10.0.0.4    HTTP      145 GET / HTTP/1.1
  176 21.64719200( 10.0.0.4   10.0.0.5    TCP        68 8000→58703 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10882466 TSecr=10882466
  177 21.64720900( 10.0.0.4   10.0.0.3    TCP        68 8000→58703 [ACK] Seq=1 Ack=78 Win=29184 Len=0 TSval=10882466 TSecr=10882466
```



```
root@sdnhubvm:~/pox[17:22] (betta)$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.5 - - [05/Nov/2022 17:31:05] "GET / HTTP/1.1" 200 -
10.0.0.5 - - [05/Nov/2022 17:31:13] "GET / HTTP/1.1" 200 -
10.0.0.5 - - [05/Nov/2022 17:31:20] "GET / HTTP/1.1" 200 -
```