

Trabajo Práctico 2 — Software Defined Networks — Grupo 12

[75.43] Introducción a los Sistemas Distribuidos
Segundo cuatrimestre de 2022

Alumnos	Padrón	Email
Codino, Federico	103533	fcodino@fi.uba.ar
De Feo, Laura	102831	ldefeo@fi.uba.ar
Demarchi, Lucas	104525	ldemarchi@fi.uba.ar
Diaz Calixto, Luz M.	105122	ldiazc@fi.uba.ar
Diaz Miguez, Abril	104956	adiazm@fi.uba.ar

Índice

1. Introducción	2
2. Hipotesis y suposiciones realizadas	2
3. Implementación	2
3.1. Topología	2
3.2. Firewall propiamente dicho	2
4. Pruebas	4
4.1. Host 1 no puede enviar como cliente por UDP al puerto 5001	4
4.2. Host 1 puede enviar como cliente por TCP al puerto 5001	4
4.3. Host 1 puede ser servidor con UDP desde el puerto 5001	5
4.4. Dos host no pueden comunicarse por el puerto 80 con TCP	6
4.5. Dos host no pueden comunicarse por el puerto 80 con UDP	6
4.6. Dos host pueden comunicarse por otro puerto con TCP	7
4.7. Dos host pueden comunicarse por otro puerto con UDP	7
4.8. Dos host definidos para estar incomunicados en distintos extremos, no se comunican	8
4.9. Dos hosts definidos incomunicados en el mismo extremo, no se comunican si el firewall es el switch que los comunica	9
4.10. Dos hosts definidos incomunicados en el mismo extremo, sí se comunican si el firewall no es el switch que los comunica	9
5. Capturas Wireshark	11
5.1. Dos host no pueden comunicarse por el puerto 80 con TCP	11
5.2. Descartar todos los mensajes que provengan del host 1, tengan como puerto destino el 5001, y estén utilizando el protocolo UDP.	14
5.3. Dos hosts cualesquiera no deben poder comunicarse de ninguna forma.	16
5.4. Dos hosts sin restricciones se pueden comunicar	19
6. Preguntas a responder	21
6.1. ¿Cuál es la diferencia entre un switch y un router? ¿Qué tienen en común?	21
6.2. ¿Cuál es la diferencia entre un Switch convencional y un Switch OpenFlow?	21
6.3. ¿Se pueden reemplazar todos los routers de la Internet por Switches OpenFlow? Piense en el escenario interesante para elaborar su respuesta	21
7. Dificultades Encontradas	21
7.1. Aprender a manejar las herramientas	21
7.2. Entender los mensajes de error de iperf	22
8. Conclusiones	22

1. Introducción

En este trabajo se implementó una topología dinámica y un firewall a nivel capa de enlace. El proyecto fue implementado en Python y en la entrega se cuenta con un archivo README que explica cómo ejecutar el programa, incluyendo las dependencias necesarias.

2. Hipotesis y suposiciones realizadas

1. Asumimos que el mensaje Connection Failed: Connection Timed Out indica que no se pudo establecer la conexión TCP.
2. Asumimos que el mensaje Connection Failed: Operation now in progress también indica que no se pudo establecer la conexión TCP.
3. Asumimos que el mensaje Warning: Did not receive ack of last datagram after 10 tries, indica que no se pudo enviar el mensaje en la conexión UDP.
4. Si se indica una posición de firewall mayor a la cantidad total de switches disponibles, ningún switch será firewall.
5. Si se envía una cantidad negativa de switches, no se agregarán más switches de los dos obligatorios en la topología.
6. No se permite que un host no pueda comunicarse consigo mismo. La tercera regla implica que dos hosts distintos pueden no comunicarse entre sí, no se permite elegir que los dos "host cualesquiera" sean el mismo host. Deben ser dos hosts distintos.

3. Implementación

3.1. Topología

Implementamos dos funciones para setear el enlace del primer switch a los dos host izquierdos y del último switch a los dos host derechos. A su vez, se implementó una estructura interna donde conectamos los switch de manera lineal según la cantidad de switch que el usuario ingrese por terminal. Esta secuencia la podemos apreciar en la siguiente imagen.

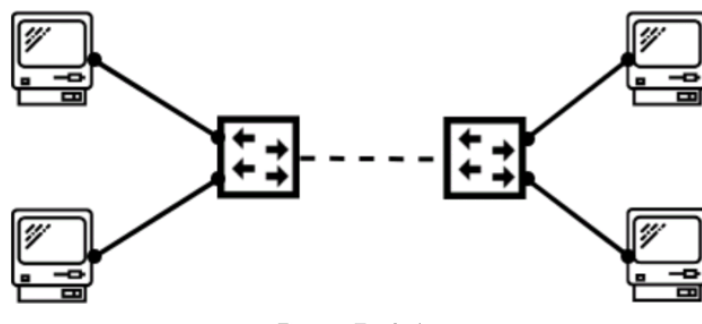


Figura 1: Topología.

3.2. Firewall propiamente dicho

Cada regla la tomamos de forma independiente, implementando su lógica en funciones separadas.

Para la regla 1 realizamos lo siguiente:

1. Como el droppeo de los mensajes que se dirigen al destino con puerto 80 funciona diferente según el protocolo que se esté utilizando, dividimos la regla en dos funciones: para el protocolo UDP y para el protocolo TCP.
2. Se instancia un mensaje con la función que provee POX denominada *ofp_flow_mod()* para poder realizar los matches pertinentes para droppear.
3. El primer match que realizamos es el *dl_type*, donde matcheamos el protocolo de la capa de red IPv4.
4. Luego, matcheamos el protocolo UDP o TCP según corresponda con la función *nw_proto*, provista por POX .
5. El último match que realizamos es el puerto de destino 80, con la función *tp_dst*.
6. Por último, enviamos el mensaje con todos los matches generados.

Para la regla 2 realizamos lo siguiente:

1. Generamos el mensaje de la misma forma antes mencionada en el punto 1.
2. Matcheamos la direccion del host 1 utilizando *dl_src* y **EthAddr("00:00:00:00:00:01")**.
3. Luego, realizamos los matches para el protocolo IPv4 y el protocolo UDP de la misma forma mencionada en los items 3 y 4 de la sección anterior.
4. El último match realizado fue el puerto de destino 5001 con la función *tp_dst*.
5. Finalmente, enviamos el mensaje con todos los matches generados.

Para la regla 3 realizamos los siguientes pasos:

1. Generar el mensaje de la misma forma que antes mencionada.
2. Asignar el host fuente que el usuario ingresó por terminal.
3. Asignar el host destino que el usuario tambien ingresó por terminal.
4. Se envía el mensaje.

Por default, los host que no se pueden comunicar son el 1 y el 4.

4. Pruebas

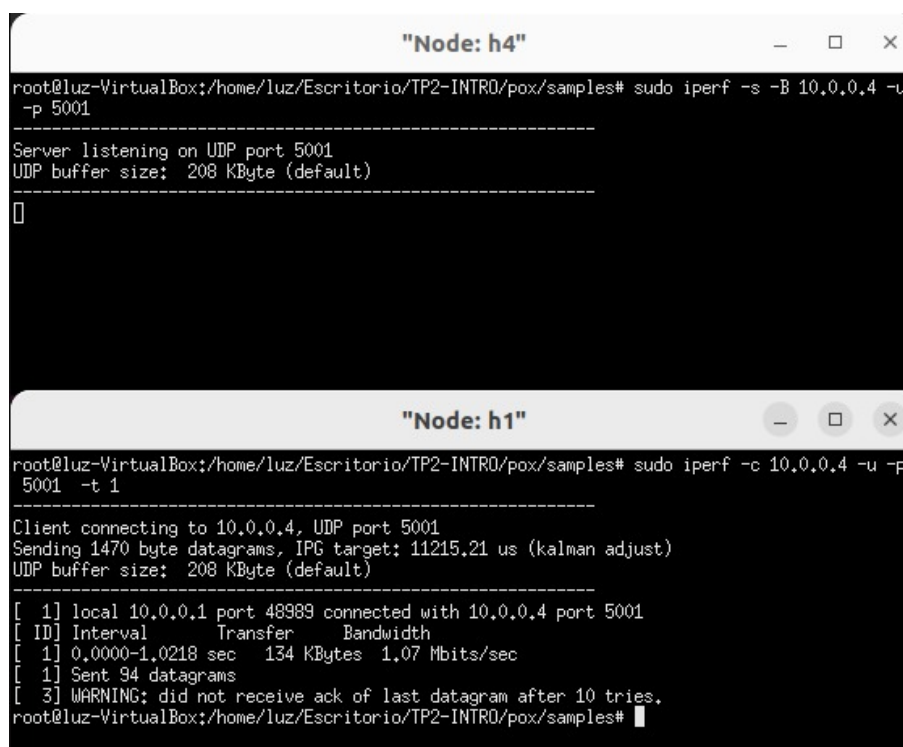
En esta sección, se detallan los diferentes casos que probamos para verificar el correcto funcionamiento del trabajo.

4.1. Host 1 no puede enviar como cliente por UDP al puerto 5001

En esta prueba se verifica el correcto funcionamiento de la regla 2.

Se levanta el pox y el mininet como se explica en el README.md, se elige al host 4 como servidor y al host 1 como cliente. Esta comunicación no debe poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:



```
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -s -B 10.0.0.4 -u -p 5001
-----
Server listening on UDP port 5001
UDP buffer size: 208 KByte (default)
-----
[]

root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -c 10.0.0.4 -u -p 5001 -t 1
-----
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.1 port 48989 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-1.0218 sec  134 KBytes  1.07 Mbits/sec
[ 1] Sent 94 datagrams
[ 3] WARNING: did not receive ack of last datagram after 10 tries.
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples#
```

Figura 2: H1 como cliente no puede enviar por UDP al puerto 5001

4.2. Host 1 puede enviar como cliente por TCP al puerto 5001

Nuevamente se eligen al host 4 como servidor y al host 1 como cliente, para que el cliente envíe por el puerto 5001 un paquete. En esta ocasión, sin embargo, se utiliza el protocolo TCP, por lo que la segunda regla no se incumple. Esta comunicación debería poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:

The image shows two terminal windows. The top window, titled "Node: h1", shows the execution of the command `sudo iperf -c 10.0.0.4 -p 5001 -t 5`. It displays the client connecting to 10.0.0.4 on TCP port 5001, and then shows a table of results for the interval 0.0000-5.0044 sec, indicating a transfer of 6.76 GBytes at a bandwidth of 11.6 Gbits/sec. The bottom window, titled "Node: h4", shows the execution of `sudo iperf -s -B 10.0.0.4 -p 5001`. It displays the server listening on TCP port 5001, and then shows a table of results for the interval 0.0000-4.9991 sec, indicating a transfer of 6.76 GBytes at a bandwidth of 11.6 Gbits/sec.

```

"Node: h1"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples# sudo iperf -c 10.0.0.4 -p 5001 -t 5
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85,3 KByte (default)
-----
[ 1] local 10.0.0.1 port 54056 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-5.0044 sec 6.76 GBytes 11.6 Gbits/sec
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples#

"Node: h4"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples# sudo iperf -s -B 10.0.0.4 -p 5001
Server listening on TCP port 5001
TCP window size: 85,3 KByte (default)
-----
[ 1] local 10.0.0.4 port 5001 connected with 10.0.0.1 port 54056
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-4.9991 sec 6.76 GBytes 11.6 Gbits/sec

```

Figura 3: H1 como cliente envía por TCP al puerto 5001

4.3. Host 1 puede ser servidor con UDP desde el puerto 5001

De la misma manera, se eligen a los host 4 y 1, pero ahora el 1 es el servidor. La comunicación se establece con UDP al puerto 5001, pero como el host 1 es el servidor, no se incumple la segunda regla. Esta comunicación debería poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:

The image shows two terminal windows. The top window, titled "Node: h1", shows the execution of the command `sudo iperf -s -B 10.0.0.1 -u -p 5001`. It displays the server listening on UDP port 5001, and then shows a table of results for the interval 0.0000-9.9931 sec, indicating a transfer of 1.28 MBytes at a bandwidth of 1.08 Mbits/sec, with 23 datagrams received out-of-order. The bottom window, titled "Node: h4", shows the execution of `sudo iperf -c 10.0.0.1 -u -p 5001`. It displays the client connecting to 10.0.0.1 on UDP port 5001, and then shows a table of results for the interval 0.0000-10.0164 sec, indicating a transfer of 1.25 MBytes at a bandwidth of 1.05 Mbits/sec, with 896 datagrams sent and 23 datagrams received out-of-order.

```

"Node: h1"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples# sudo iperf -s -B 10.0.0.1 -u -p 5001
Server listening on UDP port 5001
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.1 port 5001 connected with 10.0.0.4 port 58834
[ ID] Interval      Transfer    Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-9.9931 sec 1.28 MBytes 1.08 Mbits/sec 0.027 ms 0/895 (0%)
[ 1] 0.0000-9.9931 sec 23 datagrams received out-of-order

"Node: h4"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples# sudo iperf -c 10.0.0.1 -u -p 5001
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.4 port 58834 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0164 sec 1.25 MBytes 1.05 Mbits/sec
[ 1] Sent 896 datagrams
[ 1] Server Report:
[ ID] Interval      Transfer    Bandwidth      Jitter  Lost/Total Datagrams
[ 1] 0.0000-9.9931 sec 1.28 MBytes 1.08 Mbits/sec 0.027 ms 0/895 (0%)
[ 1] 0.0000-9.9931 sec 23 datagrams received out-of-order
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTR0/pox/samples#

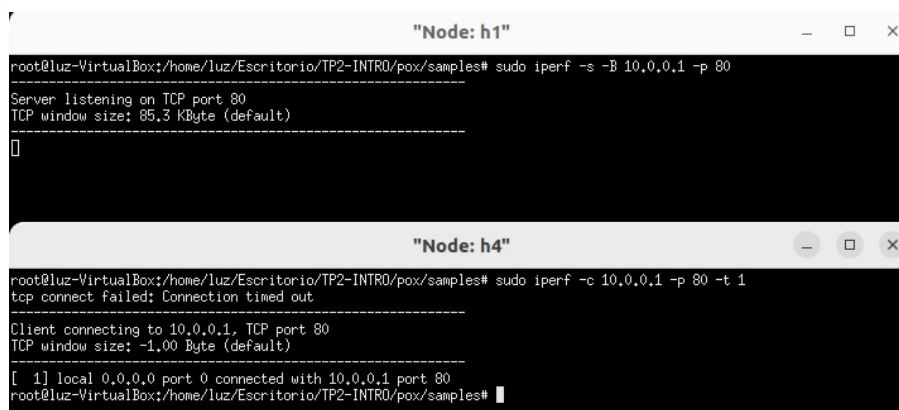
```

Figura 4: H1 como servidor envía por UDP al puerto 5001

4.4. Dos host no pueden comunicarse por el puerto 80 con TCP

Eligiendo los host 1 y 4 y estableciendo que la regla 3 no prohíba la comunicación entre ellos, cuando intentan hacerlo por TCP a través del puerto 80, la regla 1 no permite que éstos se comuniquen. Esta conexión no debería poder establecerse.

Ejecutando los comandos correctos, se puede obtener el siguiente resultado:



```
"Node: h1"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -s -B 10.0.0.1 -p 80
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
[ ]

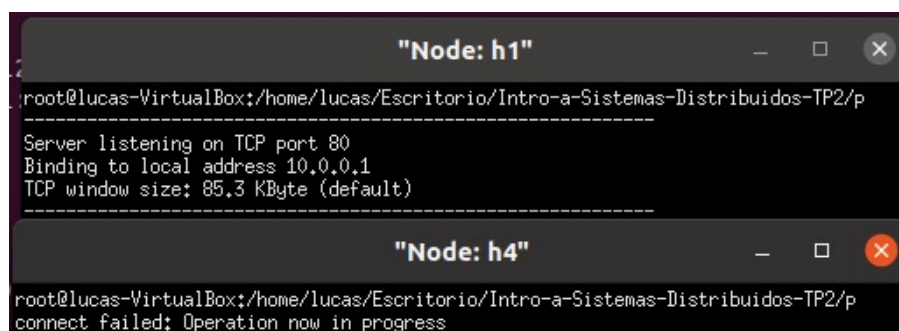
"Node: h4"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -c 10.0.0.1 -p 80 -t 1
tcp connect failed: Connection timed out

Client connecting to 10.0.0.1, TCP port 80
TCP window size: -1.00 Byte (default)

[ 1] local 0.0.0.0 port 0 connected with 10.0.0.1 port 80
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples#
```

Figura 5: H1 y H4 no se pueden comunicar por tcp puerto 80, resulta en timeout

o el siguiente resultado:



```
"Node: h1"
root@lucas-VirtualBox:/home/lucas/Escritorio/Intro-a-Sistemas-Distribuidos-TP2/p
Server listening on TCP port 80
Binding to local address 10.0.0.1
TCP window size: 85.3 KByte (default)
-----

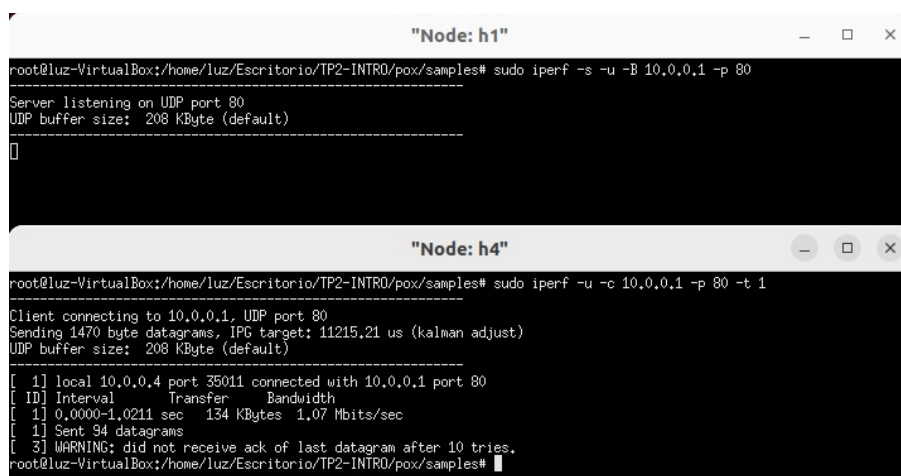
"Node: h4"
root@lucas-VirtualBox:/home/lucas/Escritorio/Intro-a-Sistemas-Distribuidos-TP2/p
connect failed: Operation now in progress
```

Figura 6: H1 y H4 no se pueden comunicar por tcp puerto 80, resulta en Operation Now In Progress

4.5. Dos host no pueden comunicarse por el puerto 80 con UDP

Eligiendo los host 1 y 4 y estableciendo que la regla 3 no prohíba la comunicación entre ellos, cuando intentan hacerlo por UDP a través del puerto 80, la regla 1 no permite que éstos se comuniquen. Esta conexión no debería poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:



The image shows two terminal windows. The top window, titled "Node: h1", shows a server listening on UDP port 80. The bottom window, titled "Node: h4", shows a client connecting to 10.0.0.1 on UDP port 80. The client sends 1470 byte datagrams with an IPG target of 11215.21 us. The output shows a successful connection to local 10.0.0.4 port 35011, with a transfer of 134 KBytes at 1.07 Mbits/sec over an interval of 0.0000-1.0211 sec. A warning is displayed: "WARNING: did not receive ack of last datagram after 10 tries."

```
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -s -u -B 10.0.0.1 -p 80
Server listening on UDP port 80
UDP buffer size: 208 KByte (default)

[Node: h1]

root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -c 10.0.0.1 -p 80 -t 1
Client connecting to 10.0.0.1, UDP port 80
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)

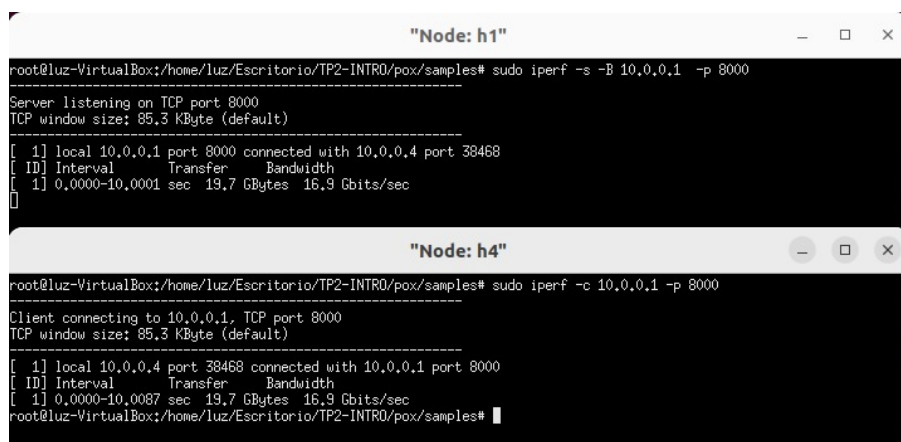
[ 1] local 10.0.0.4 port 35011 connected with 10.0.0.1 port 80
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-1.0211 sec  134 KBytes  1.07 Mbits/sec
[ 1] Sent 94 datagrams
[ 3] WARNING: did not receive ack of last datagram after 10 tries.
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples#
```

Figura 7: H1 y H4 no se pueden comunicar por udp puerto 80

4.6. Dos host pueden comunicarse por otro puerto con TCP

Eligiendo los host 1 y 4 y estableciendo que la regla 3 no prohíba la comunicación entre ellos, cuando intentan hacerlo por TCP a través de un puerto no prohibido, por ejemplo el puerto 8000, la regla 1 no prohíbe que éstos se comuniquen. Esta conexión debería poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:



The image shows two terminal windows. The top window, titled "Node: h1", shows a server listening on TCP port 8000. The bottom window, titled "Node: h4", shows a client connecting to 10.0.0.1 on TCP port 8000. The output shows a successful connection to local 10.0.0.4 port 38468, with a transfer of 19.7 GBytes at 16.9 Gbits/sec over an interval of 0.0000-10.0087 sec.

```
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -s -B 10.0.0.1 -p 8000
Server listening on TCP port 8000
TCP window size: 85.3 KByte (default)

[Node: h1]

root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -c 10.0.0.1 -p 8000
Client connecting to 10.0.0.1, TCP port 8000
TCP window size: 85.3 KByte (default)

[ 1] local 10.0.0.4 port 38468 connected with 10.0.0.1 port 8000
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0087 sec  19.7 GBytes  16.9 Gbits/sec
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples#
```

Figura 8: H1 y H4 se pueden comunicar por tcp puerto 8000

4.7. Dos host pueden comunicarse por otro puerto con UDP

Eligiendo los host 1 y 4 y estableciendo que la regla 3 no prohíba la comunicación entre ellos, cuando intentan hacerlo por UDP a través de un puerto no prohibido, por ejemplo el puerto 8000, la regla 1 no prohíbe que éstos se comuniquen. Esta conexión debería poder establecerse.

Ejecutando los comandos correctos, se obtiene el siguiente resultado:


```

"Node: h1"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -s -B 10.0.0.1 -u -p 8000
Server listening on UDP port 8000
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.1 port 8000 connected with 10.0.0.4 port 40459
[ ID] Interval      Transfer    Bandwidth      Jitter   Lost/Total Datagrams
[ 1] 0.0000-9.9814 sec 1.29 MBytes 1.09 Mbits/sec 0.022 ms 0/895 (0%)
[ 1] 0.0000-9.9814 sec 31 datagrams received out-of-order

"Node: h4"
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples# sudo iperf -c 10.0.0.1 -u -p 8000
Client connecting to 10.0.0.1, UDP port 8000
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 1] local 10.0.0.4 port 40459 connected with 10.0.0.1 port 8000
[ ID] Interval      Transfer    Bandwidth      Jitter   Lost/Total Datagrams
[ 1] 0.0000-10.0154 sec 1.25 MBytes 1.05 Mbits/sec
[ 1] Sent 896 datagrams
[ 1] Server Report:
[ ID] Interval      Transfer    Bandwidth      Jitter   Lost/Total Datagrams
[ 1] 0.0000-9.9814 sec 1.29 MBytes 1.09 Mbits/sec 0.021 ms 0/895 (0%)
[ 1] 0.0000-9.9814 sec 31 datagrams received out-of-order
root@luz-VirtualBox:/home/luz/Escritorio/TP2-INTRO/pox/samples#

```

Figura 9: H1 y H4 se pueden comunicar por udp puerto 8000

4.8. Dos host definidos para estar incomunicados en distintos extremos, no se comunican

Se eligen los hosts 2 y 3 y se envía su número por comando en la terminal. Cuando se levanta a uno como el servidor y a otro como el cliente, el paquete sí o sí pasa por el firewall, por lo que no es posible comunicarlos.

Al ejecutar los comandos correctos, se obtiene el siguiente resultado, donde el comando pingall denota que los hosts 2 y 3 no se logran comunicar:

```

luz@luz-VirtualBox:~/Escritorio/TP2-INTRO$ python3 pox.py log.level --DEBUG openflow.of_01 forwarding.l2_learning samples.custom_firewall --firewallPosition=2 --host_not_src=2 --host_not_dst=3
luz@luz-VirtualBox:~/Escritorio/TP2-INTRO/pox/samples$ sudo mn --custom ./custom_topology.py --topo mytopo,5 --mac --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6) (s6, h3) (s6, h4)

*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 X h4
h3 -> h1 X h4
h4 -> h1 h2 h3
*** Results: 16% dropped (10/12 received)

```

Figura 10: H2 y H3 están en diferentes extremos, por lo que la regla 1 se aplica siempre

4.9. Dos hosts definidos incomunicados en el mismo extremo, no se comunican si el firewall es el switch que los comunica

Se eligieron los hosts H1 y H2 para demostrar el comportamiento. Se desea que ambos hosts estén incomunicados y se sabe que entre ellos hay un solo switch, el del extremo izquierdo.

Si se elige a ese switch como firewall, se aplica la regla y se obtiene el siguiente resultado

```
luz@luz-VirtualBox:~/Escritorio/TP2-INTRO$ python3 pox.py log.level --DEBUG op
enflow.of_01 forwarding.l2_learning samples.custom_firewall --firewallPosition
=1 --host not src=1 --host not dst=2
luz@luz-VirtualBox:~/Escritorio/TP2-INTRO/pox/samples$ sudo mn --custom ./custom
_topology.py --topo mytopo,5 --mac --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6) (s6, h3) (s6, h4)

*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4
h2 -> X h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 16% dropped (10/12 received)
mininet>
```

Figura 11: H1 y H2 están en el mismo extremos con el firewall en el switch que los comunica. La regla 1 se aplica.

Es decir, los dos hosts están incomunicados.

4.10. Dos hosts definidos incomunicados en el mismo extremo, sí se comunican si el firewall no es el switch que los comunica

En cambio, si se elige otro switch como firewall, por ejemplo, el segundo en la topología lineal de 7 switches, el paquete no deberá pasar por ese switch para llegar de un host a otro. No se aplicará la regla. Eligiendo el switch 2, el resultado es el siguiente:

```
luz@luz-VirtualBox:~/Escritorio/TP2-INTRO$ python3 pox.py log.level --DEBUG op
enflow.of_01 forwarding.l2_learning samples.custom_firewall --firewallPosition
=2 --host_not_src=1 --host_not_dst=2
luz@luz-VirtualBox:~/Escritorio/TP2-INTRO/pox/samples$ sudo mn --custom ./custom
_topology.py --topo mytopo,5 --mac --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6) (s6, h3) (s6, h4)

*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Figura 12: H1 y H2 están en el mismo extremo con el firewall en un switch que no los comunica. La regla 1 no se aplica.

Es decir, los dos hosts pueden comunicarse, aunque el usuario haya elegido que esos dos no se comuniquen, debido a la posición del firewall en la topología.

5. Capturas Wireshark

En esta sección se muestran las capturas de las pruebas que consideramos más relevantes obtenidas en wireshark.

5.1. Dos host no pueden comunicarse por el puerto 80 con TCP

A continuación se muestran las capturas realizadas cuando dos hosts distintos tratan de comunicarse por el puerto 80 con protocolo TCP.

Las siguientes dos imagenes muestran los mensajes detectados en la red localhost. Allí podemos observar que se envían los mensajes TCP y los del protocolo OpenFlow.

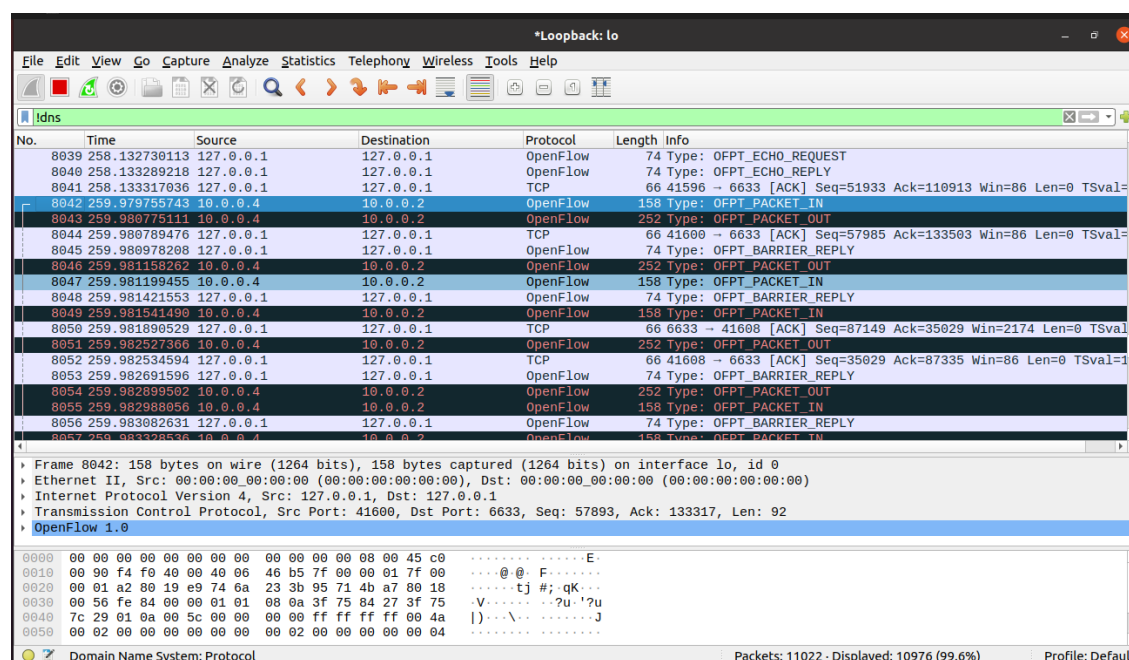


Figura 13: Mensajes detectados en la red localhost Parte 1

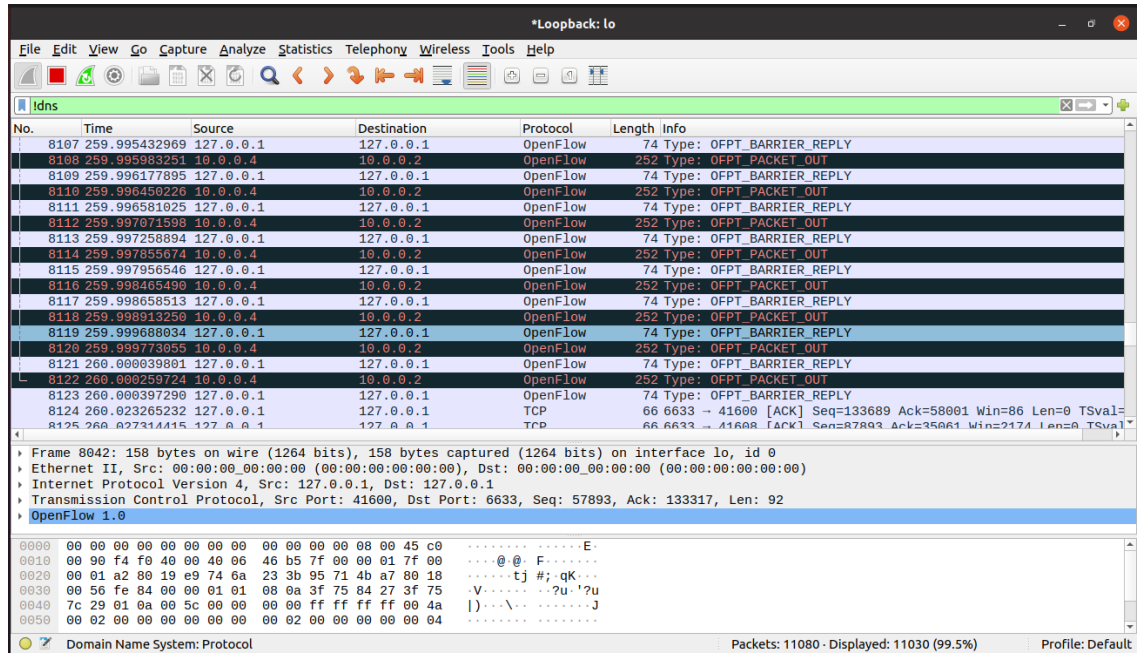


Figura 14: Mensajes detectados en la red localhost Parte 2

Luego se decidió obtener las capturas de lo que ocurría dentro del switch que actúa como firewall, en este caso el switch S2, para observar su comportamiento. Primero se muestran las capturas de la entrada del switch. En ésta, se observan que llegan los mensajes SYN de TCP con puerto destino 80.

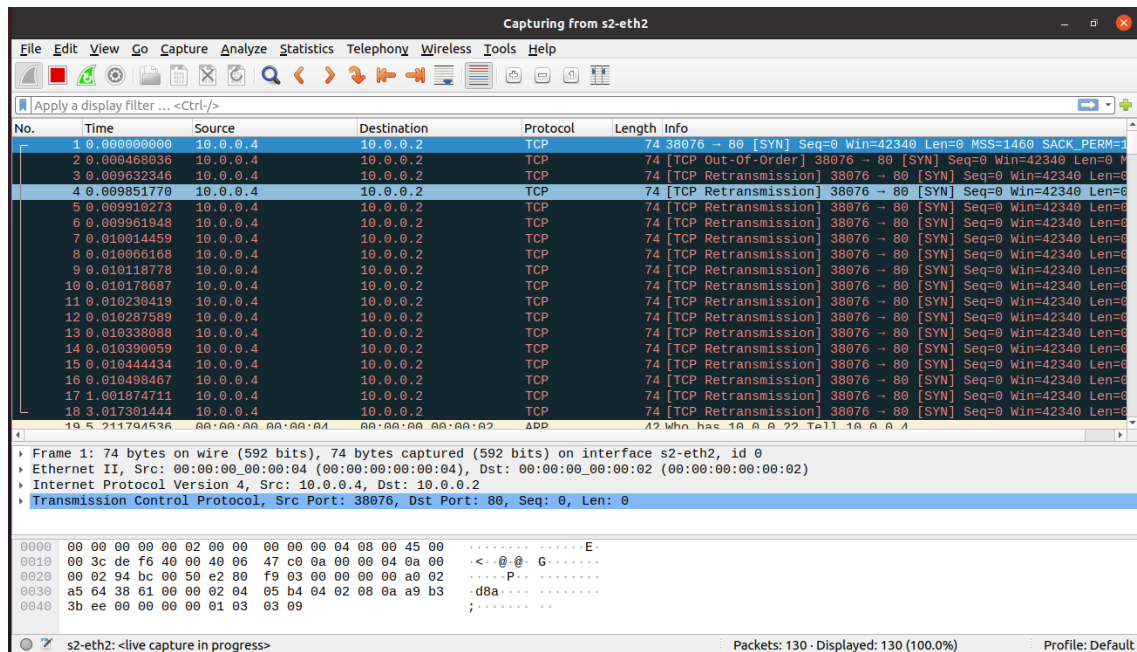


Figura 15: Mensajes detectados en la entrada del switch que hace de firewall (S2) Parte 1

A continuación, se muestra la captura del mensaje de protocolo ARP, para que el switch cliente aprenda dónde se encuentra el servidor y guardarlo en su tabla.

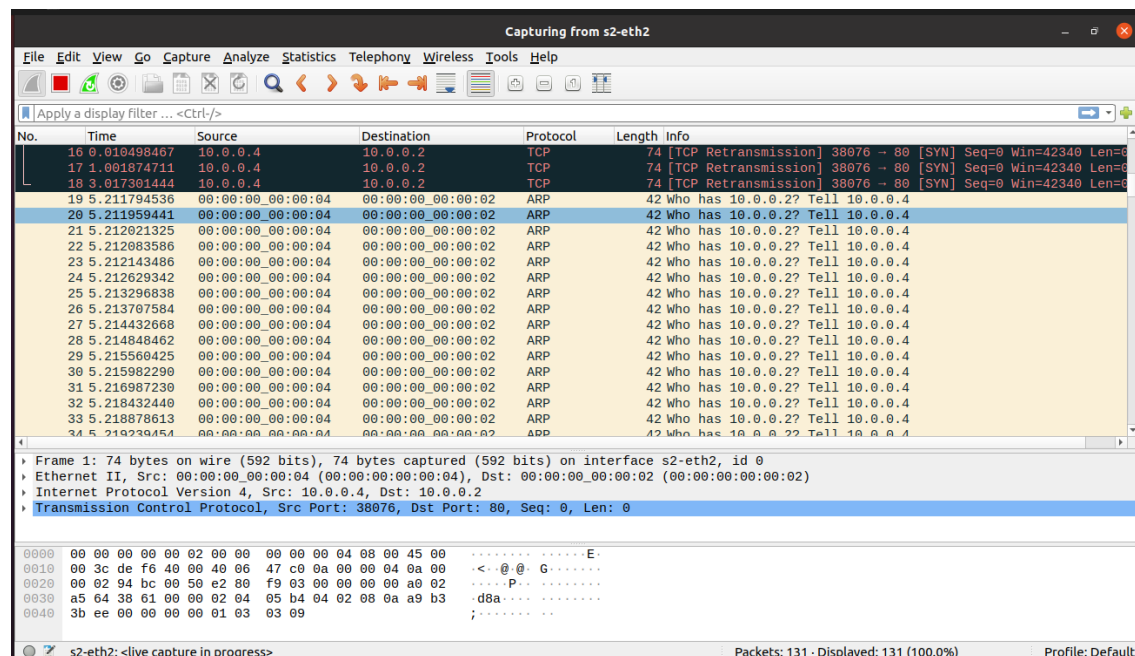


Figura 16: Mensajes detectados en la entrada del switch que hace de firewall (S2) Parte 2

Por último se muestra la captura de la salida del switch, en la cual se observa que no salen los mensajes del tipo SYN TCP con puerto destino 80.

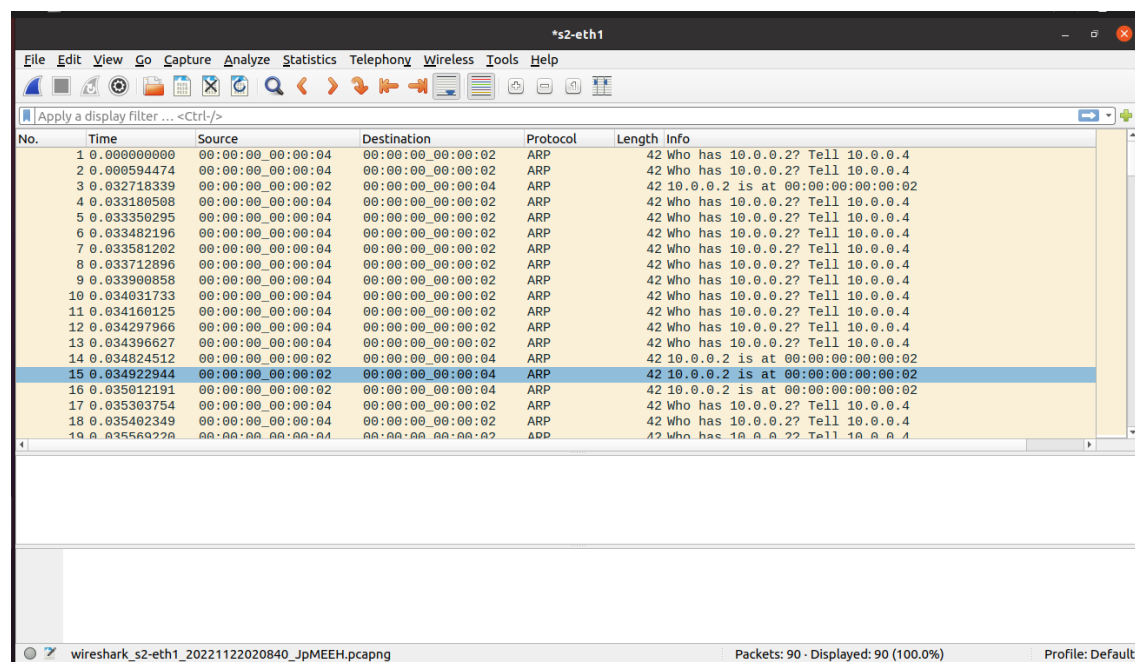


Figura 17: Mensajes detectados en la salida del switch que hace de firewall (S2)

Esto implica que no se terminó de establecer la conexión TCP entre los dos hosts.

5.2. Descartar todos los mensajes que provengan del host 1, tengan como puerto destino el 5001, y estén utilizando el protocolo UDP.

Las imágenes que se muestran a continuación, ejemplifican qué sucede cuando el host 1 intenta mandar un mensaje UDP con puerto destino 5001.

Al igual que en el caso anterior, la primera imagen muestra los mensajes detectados en la red localhost. Allí podemos observar que se envían los mensajes del protocolo OpenFlow.

Capturing from Loopback: lo

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
2592	8.947425906	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2593	8.947939375	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2594	8.948075093	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2595	8.948484815	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2596	8.948617606	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2597	8.950144995	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2598	8.950312216	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2599	8.950807019	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2600	8.950957714	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2601	8.951565390	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2602	8.951716184	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2603	8.952210952	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2604	8.952362665	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2605	8.953190467	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2606	8.953381456	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2607	8.953885183	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2608	8.954488009	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY
2609	8.954538154	00:00:00:00:00:03	00:00:00:00:00:01	OpenFlow	220	Type: OFPT_PACKET_OUT
2610	8.954578577	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_BARRIER_REPLY

▶ Frame 2602: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0

▶ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

▶ Transmission Control Protocol, Src Port: 34378, Dst Port: 6633, Seq: 15609, Ack: 62387, Len: 8

▶ OpenFlow 1.0

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figura 18: Mensajes detectados en la red localhost

Luego, al igual que en el caso anterior, se decidió obtener las capturas de lo que ocurría dentro del switch que actúa como firewall, en este caso el switch S2, para observar su comportamiento. Primero se muestran las capturas de la entrada del switch, en las cuales se observan que llegan los mensajes UDP con puerto destino 5001 de la dirección 10.0.0.1(host 1).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
2	0.000590530	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
3	0.009823740	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
4	0.021115012	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
5	0.032043741	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
6	0.043457618	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
7	0.054604160	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
8	0.065822672	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
9	0.076850366	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
10	0.088233801	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
11	0.099458508	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
12	0.110671988	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
13	0.121891287	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
14	0.133012602	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
15	0.144369790	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
16	0.155536211	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
17	0.166810401	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
18	0.178012849	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
19	0.189247185	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470

Frame 11: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits) on interface s2-eth1, id 0
 Ethernet II, Src: 00:00:00:00:00:01 (00:00:00:00:00:01), Dst: 00:00:00:00:00:03 (00:00:00:00:00:03)
 Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.3
 User Datagram Protocol, Src Port: 48502, Dst Port: 5001
 Data (1470 bytes)

0000 00 00 00 00 00 03 00 00 00 00 01 08 00 45 00E..
 0010 05 da c4 93 40 00 40 11 5c 7c 0a 00 00 01 0a 00@.@.\|.....
 0020 00 03 bd 76 13 89 05 c6 19 db 00 00 01 54 63 7c ...v.....Tc|
 0030 62 24 00 0c f9 e5 00 00 00 08 00 00 00 30 31 b\$......01
 0040 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 23456789 01234567
 0050 38 39 30 31 00 34 34 35 36 37 38 39 30 31 32 33 8901.445 67890123

Figura 19: Mensajes detectados en la entrada del switch que hace de firewall (S2) Parte 1

No.	Time	Source	Destination	Protocol	Length	Info
446	4.977790396	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
447	4.989025870	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
448	5.000211885	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
449	5.130580260	00:00:00:00:00:01	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
450	5.131895569	00:00:00:00:00:01	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
451	5.250510107	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
452	5.500710359	10.0.0.1	10.0.0.3	UDP	1512	48502 → 5001 Len=1470
453	5.534483319	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
454	5.535077214	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
455	5.577088742	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
456	5.577234659	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
457	5.577404367	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
458	5.577522939	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
459	5.577713264	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
460	5.577840128	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
461	5.578016417	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
462	5.578152174	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
463	5.578332182	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
464	5.578465529	00:00:00:00:00:03	00:00:00:00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03

Frame 342: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits) on interface s2-eth1, id 0
 Ethernet II, Src: 00:00:00:00:00:01 (00:00:00:00:00:01), Dst: 00:00:00:00:00:03 (00:00:00:00:00:03)
 Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.3
 User Datagram Protocol, Src Port: 48502, Dst Port: 5001
 Data (1470 bytes)

0000 00 00 00 00 00 03 00 00 00 00 01 08 00 45 00E..
 0010 05 da c4 93 40 00 40 11 5c 7c 0a 00 00 01 0a 00@.@.\|.....
 0020 00 03 bd 76 13 89 05 c6 19 db 00 00 01 54 63 7c ...v.....Tc|
 0030 62 24 00 0c f9 e5 00 00 00 08 00 00 00 30 31 b\$......01
 0040 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 23456789 01234567
 0050 38 39 30 31 00 34 34 35 36 37 38 39 30 31 32 33 8901.445 67890123

Figura 20: Mensajes detectados en la entrada del switch que hace de firewall (S2) Parte 2

Por último, en la figura 21 se muestra la captura de la salida del switch en el cual se observa que no salen los mensajes del tipo UDP con puerto destino 5001 con origen en el host 1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:00:00_00:00:01	00:00:00_00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
2	0.009304472	00:00:00_00:00:01	00:00:00_00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
3	0.009421365	00:00:00_00:00:01	00:00:00_00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
4	0.009487700	00:00:00_00:00:01	00:00:00_00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.1
5	0.188734947	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
6	0.189261582	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
7	0.189707224	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
8	0.189801048	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
9	0.190282174	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
10	0.191059564	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
11	0.191150375	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
12	0.191598222	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
13	0.192126538	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
14	0.192266687	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
15	0.192691096	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
16	0.193097720	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
17	0.193499680	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
18	0.193586400	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
19	0.194357308	00:00:00_00:00:03	00:00:00_00:00:01	ARP	42	10.0.0.3 is at 00:00:00:00:00:03

Frame 454: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface s2-eth2, id 0
 Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: 00:00:00_00:00:01 (00:00:00:00:00:01)
 Address Resolution Protocol (reply)

```

0000  00 00 00 00 00 01 00 00 00 00 03 08 06 00 01  .....
0010  08 00 06 04 00 02 00 00 00 00 03 0a 00 00 03  .....
0020  00 00 00 00 00 01 0a 00 00 01  .....

```

s2-eth2: <live capture in progress> Packets: 836 - Displayed: 836 (100.0%) Profile: Default

Figura 21: Mensajes detectados en la salida del switch que hace de firewall (S2)

5.3. Dos hosts cualesquiera no deben poder comunicarse de ninguna forma.

A continuación, se muestran las capturas donde los host h3 y h2 no pueden comunicarse entre sí. Para evaluar este caso, se realizó un ping desde h2 a h3 y se capturó su comportamiento.

La imagen siguiente muestra los mensajes detectados en la red localhost. Allí podemos observar que se envían los mensajes del protocolo OpenFlow.



Luego, al igual que en los casos anteriores, se decidió obtener las capturas de lo que ocurría dentro del switch que actúa como firewall, en este caso S2, para observar su comportamiento. Por esto, en las figuras 23 y 24 se muestran los mensajes ICMP con fuente 10.0.0.2(host 2) y destino 10.0.0.3(host 3).



No.	Time	Source	Destination	Protocol	Length	Info
402	35.839413672	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x7593, seq=36/9216, ttl=64 (no res
403	68.020182527	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=1/256, ttl=64 (no respo
404	68.021713263	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=1/256, ttl=64 (no respo
405	69.023780321	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=2/512, ttl=64 (no respo
406	70.051432346	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=3/768, ttl=64 (no respo
407	71.071093775	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=4/1024, ttl=64 (no respo
408	72.095132159	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=5/1280, ttl=64 (no respo
409	73.119366810	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=6/1536, ttl=64 (no respo
410	73.122793653	00:00:00:00:00:02	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
411	73.123687814	00:00:00:00:00:02	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
412	74.143365261	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=7/1792, ttl=64 (no respo
413	74.143551380	00:00:00:00:00:02	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
414	75.167150004	10.0.0.2	10.0.0.3	ICMP	98	Echo (ping) request id=0x75aa, seq=8/2048, ttl=64 (no respo
415	75.171357043	00:00:00:00:00:02	00:00:00:00:00:03	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
416	76.192695184	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
417	76.235202339	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
418	76.319470392	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
419	76.319666052	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
420	76.319787000	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface s2-eth1, id 0
 Ethernet II, Src: 00:00:00:00:00:02 (00:00:00:00:00:02), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Address Resolution Protocol (request)

0000 ff ff ff ff ff 00 00 00 02 08 06 00 01
 0010 08 00 06 04 00 01 00 00 00 02 0a 00 00 02
 0020 00 00 00 00 00 00 0a 00 00 03

Figura 24: Mensajes detectados en la entrada del switch que hace de firewall (S2) Parte 2

Por último, se muestra la captura de la salida del switch en el cual se observa que no viajan los mensajes del tipo ICMP hacia 10.0.0.3 (host 3).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
2	0.000041625	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
3	0.003291787	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
4	0.003620516	00:00:00:00:00:02	Broadcast	ARP	42	Who has 10.0.0.3? Tell 10.0.0.2
5	0.008855554	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
6	0.009321602	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
7	0.009944337	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
8	0.100362317	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
9	0.101308637	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
10	0.101808174	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
11	0.102876297	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
12	0.102948045	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
13	0.103260201	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
14	0.103940813	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
15	0.104591765	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
16	0.104938036	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
17	0.105594036	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
18	0.106183667	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03
19	0.106654228	00:00:00:00:00:03	00:00:00:00:00:02	ARP	42	10.0.0.3 is at 00:00:00:00:00:03

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface s2-eth2, id 0
 Ethernet II, Src: 00:00:00:00:00:02 (00:00:00:00:00:02), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Address Resolution Protocol (request)

0000 ff ff ff ff ff 00 00 00 02 08 06 00 01
 0010 08 00 06 04 00 01 00 00 00 02 0a 00 00 02
 0020 00 00 00 00 00 00 0a 00 00 03

Figura 25: Mensajes detectados en la salida del switch que hace de firewall (S2)

5.4. Dos hosts sin restricciones se pueden comunicar

En esta sección, se mostrarán las capturas de dos hosts que pueden comunicarse libremente, con el fin de contrastar el comportamiento con los casos anteriores. Para demostrar el siguiente punto se realizó un ping desde 10.0.0.1 (host 1) hacia 10.0.0.4 (host 4).

La imagen siguiente muestra los mensajes detectados en la red localhost. Allí podemos observar que se envían los mensajes del protocolo OpenFlow.

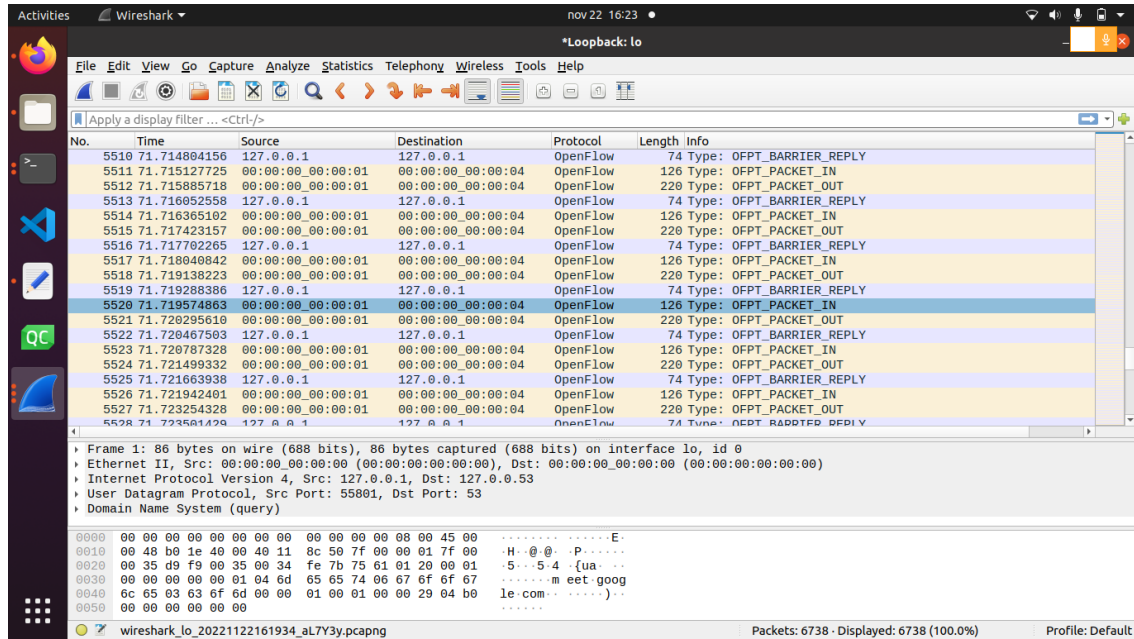


Figura 26: Mensajes detectados en la red localhost

Luego, al igual que en los casos anteriores, se decidió obtener las capturas de lo que ocurría dentro del switch que actúa como firewall, en este caso S2, para observar su comportamiento. Nuevamente, se muestra la captura de la entrada del switch, donde se observa que llegan los mensajes ICMP con fuente 10.0.0.1 (host 1) y con destino 10.0.0.4 (host 4).

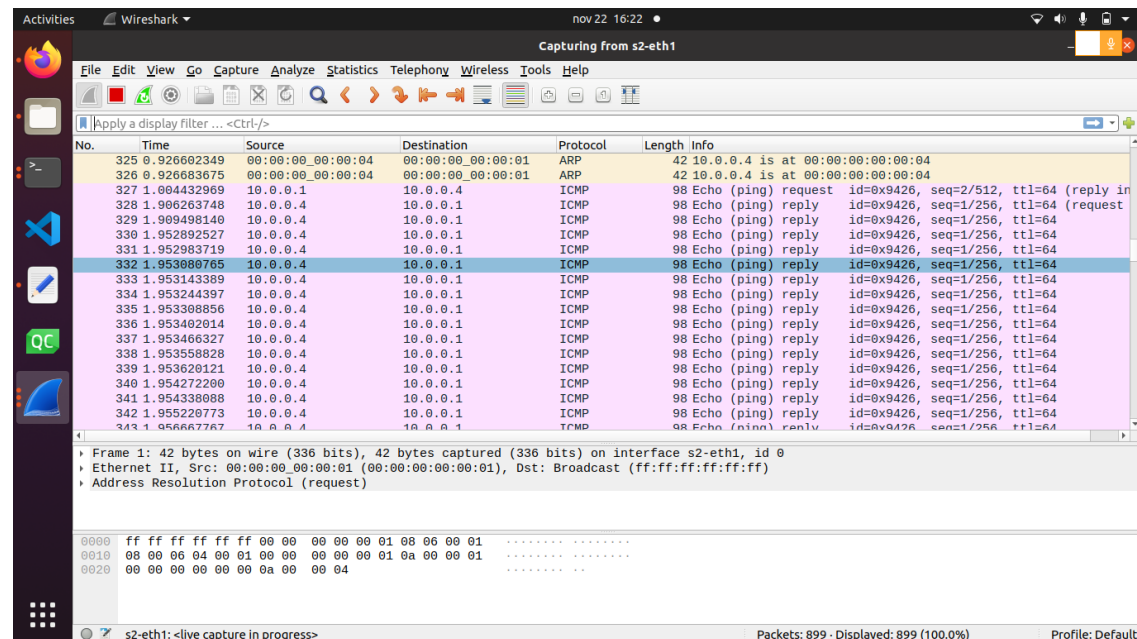


Figura 27: Mensajes detectados en la entrada del switch que hace de firewall (S2)

Por último, se muestra la captura de la salida del switch en el cual se observa que viajan los mensajes del tipo ICMP hacia 10.0.0.4 (host 4).

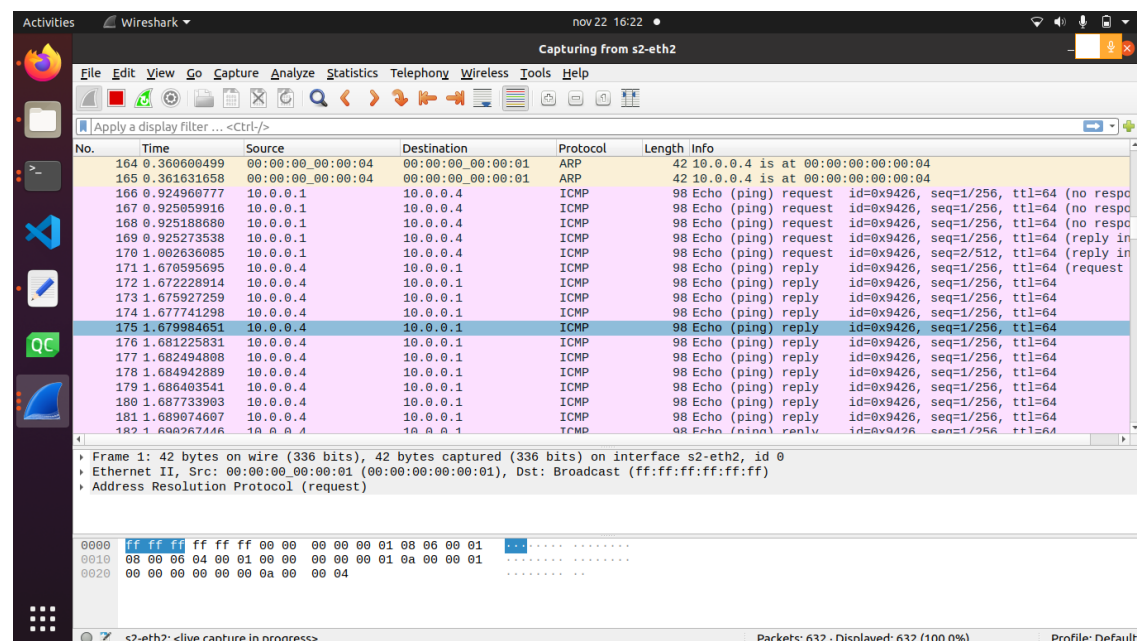


Figura 28: Mensajes detectados en la salida del switch que hace de firewall (S2)

6. Preguntas a responder

6.1. ¿Cuál es la diferencia entre un switch y un router? ¿Qué tienen en común?

Tanto el router como el switch son dispositivos de conexión utilizados en redes. Además, ambos se utilizan para redireccionar paquetes, solo que el router lo hace en capa de red, mientras que el switch lo hace en capa de enlace.

La principal diferencia entre switch y router es que el switch se encarga de crear una red conectando diferentes dispositivos entre ellos y el router establece la conexión entre estas redes. Un router conecta **diferentes redes** entre sí, mientras que un switch conecta **múltiples dispositivos** entre sí para crear una red.

Otra de las diferencias que tienen estos dispositivos es el propósito principal de cada uno de ellos. Los routers se encargan de determinar **la mejor ruta y la más corta** para que un paquete arribe a destino, mientras que los switches **reciben** un paquete de datos, lo **procesan** con el fin de determinar su dirección de destino y lo **reenvían** a esa dirección revelada.

6.2. ¿Cuál es la diferencia entre un Switch convencional y un Switch OpenFlow?

Un switch normal trabaja de forma independiente al resto de la red, tanto el forwarding y el ruteo de alto nivel son responsabilidad del switch. Un switch convencional contiene la lógica para implementar tanto el control como el dataplane.

Un switch OpenFlow, cuando recibe un paquete para el cual no tiene un flujo, contacta al controlador SDN y le pregunta qué debería hacer con este paquete. Un switch OpenFlow sólo contiene la lógica para implementar el dataplane; la lógica del control plane se encuentra en el controlador SDN.

6.3. ¿Se pueden reemplazar todos los routers de la Internet por Switches OpenFlow? Piense en el escenario interesante para elaborar su respuesta

Tal vez es posible, pero no se trata de una buena idea. Las principales razones son las siguientes:

- Se requiere el reemplazo de una gran cantidad de equipos, lo cual no es económico.
- No se ha probado en redes grandes o pruebas reales de uso.
- Los routers de borde están diseñados para la respuesta rápida en base a la IP. Openflow no trabaja con direcciones IP's, sino que trabaja con reglas como las mencionadas en el código del archivo *custom_firewall.py*. Esto provocará tablas gigantes que deberán ser consultadas para lograr el tráfico de internet.
- Es susceptible a ataques denominados **Man in the middle**.

7. Dificultades Encontradas

7.1. Aprender a manejar las herramientas

Aprender a manejar las herramientas, especialmente pox, fue de las mayores dificultades que enfrentamos en este trabajo. Pasamos una gran parte del tiempo dedicado al trabajo investigando en la documentación oficial las diferentes funciones y clases, para aprender a utilizarlas.

7.2. Entender los mensajes de error de iperf

Cuando probamos los casos de error de iperf, resultaba confuso que al enviar paquetes por TCP, apareciese el mensaje de error *Connection Failed: Operation now in progress*. En comparación, nos pareció que el mismo error pero con UDP era mucho más claro: *Warning: Did not receive ack of last datagram after 10 tries*.

A su vez, nos confundió que la conexión por TCP y UDP no se estableciese, como esperábamos, pero que el comando pingall aún así lograra enviar los paquetes. Más tarde, comprendimos que se debía a que pingall no usa el puerto que nosotros estábamos probando, por lo que era esperable que pingall funcionase cuando solamente estuviésemos filtrando por un puerto específico.

8. Conclusiones

Podemos concluir que los Switches OpenFlow son muy prometedores permitiendo que la lógica se encuentre en un controlador central, más sencillo de mantener y modificar. Con eso dicho, se debe tener en cuenta todo lo mencionado en la pregunta 3, además del hecho de que pueden introducir un cuello de botella en grandes y exigentes redes, donde todos los switches deben conectarse al controlador por cada paquete que no saben reenviar.

Es cierto que la herramienta resulta muy útil para fines didácticos, proveyendo una interfaz para simular redes de manera performante y permitir experimentar con diferentes topologías.