FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Trabajo Práctico 2 - SDN

Introducción a los Sistemas Distribuidos (75.43)

Grupo 1

Andrés Fernández - 102220
Sebastián Capelli - 98316
Francisco Bertolotto - 102671
Agustín López Núñez - 101826
Mauro Santoni - 102654

Introducción

El objetivo de este trabajo práctico es establecer una red virtual configurable, y dentro de esa red modificar las tablas de ruteo de uno de los switches, que actuará como firewall, de forma tal de bloquear la comunicación entre dos o más flujos en base a la combinación de algunos campos de los paquetes (host, puerto o protocolo).

A lo largo de este informe, se buscará explicar acerca de las herramientas utilizadas para lograr dicho objetivo, sobre las características de las reglas de ruteo establecidas en base a los distintos parámetros, y se detallarán las pruebas realizadas para verificar el correcto funcionamiento de dichas reglas.

Hipótesis y supuestos

- Los parámetros de las reglas por defecto pueden ser modificados, siendo éstos el puerto, la IP o el protocolo de capa de transporte.
- Con excepción de las reglas custom, utilizaremos el componente de pox *forwarding.l2_learning* para configurar las tablas openflow de los switches de una forma equivalente a la que se configurarían los link-layer switches.
- En el apartado Hipótesis sobre reglas de bloqueo, se detallan algunas hipótesis tomadas acerca de ciertas reglas de ruteo.

Herramientas utilizadas

Mininet

Mininet es una herramienta que permite emular una red completa de hosts, enlaces y switches en un mismo ordenador.
Permite personalizar la red en cuanto a la cantidad de hosts y switches, y en cuanto a las conexiones que existen entre hosts y switches, y entre los mismos switches.

Mininet Topology Visualizer

Permite generar gráficos sobre la topología de la red desplegada en mininet. El funcionamiento es muy sencillo ya que simplemente se deben pasar los resultados de los comandos *dump* y *link*, ejecutados en mininet, al utilitario online.

Protocolo OpenFlow

Permite la definición de políticas de ruteo de los paquetes de la red por medio de un controlador central. El controlador se implementa por software con las reglas requeridas y estas se envían hacia los switches por medio de una comunicación OpenFlow, de forma tal que se pueda determinar la acción del switch para cada flujo.
El protocolo estipula la separación data plane/control plane y los tipos de mensajes enviados entre switches y el controlador.

POX

POX es una herramienta que implementa el protocolo OpenFlow y permite configurar y ejecutar un controlador para la configuración remota de las tablas de flujo de los switches.
En este trabajo, se utilizó como complemento de Mininet, para poder trasladar las reglas solicitadas a los switches de la red mininet.

Wireshark

Wireshark es un sniffer. Muy adecuado para estudiar tráfico de paquetes y sus protocolos. En este trabajo permitió observar y estudiar los paquetes en tráfico al realizar las simulaciones con mininet + pox.

Iperf

Se trata de una herramienta que permite realizar medidas estadísticas sobre el desempeño de la red y la comunicación entre hosts. En particular, para este trabajo, resultó útil para el testeo de la comunicación, entre hosts de mininet, de ciertos flujos predeterminados.

Implementación

Sobre la topología de la red

De acuerdo al enunciado de este trabajo, la topología de la red virtual a emular tendrá las siguientes caracteristicas:

- La cantidad de hosts está predefinida, así como su conexión con los switches: se tendrán dos hosts en cada extremo de la red, y cada par de hosts se conectará a un único switch: h1 y h2 se conectarán siempre al primer switch de la red, mientras que h3 y h4 se conectarán siempre al último switch.
- La conexión entre switches será en secuencia por medio de un único enlace.

La **figura 1** muestra un ejemplo de una red que cumple estas características. En donde se cuentan con 4 switches

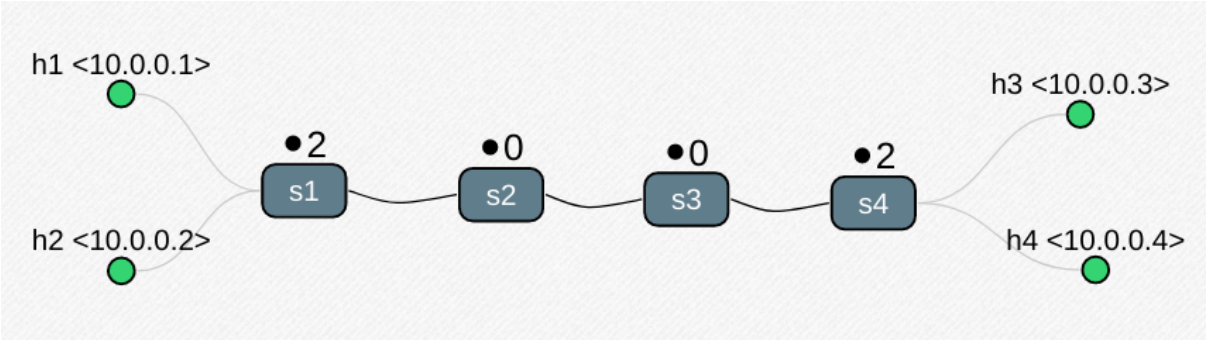


Figura 1. Topología
Link al utilitario **Mininet Topology Visualizer** de Spear by Narmox

¿Cómo se define la topología?

Mientras que la ejecución básica de mininet es con el comando *sudo mn*. en nuestro caso, la herramienta se ejecuta con los siguientes parámetros:

```
sudo mn --custom ./topology.py --topo custom,4 --mac --controller remote
```

Los parámetros adicionales se utilizan para lo siguiente:

`--custom ./topology.py`: permite personalizar la red a crear a partir de un archivo de Python (en este caso, el archivo *topology.py*). Este archivo debe contener una clase que herede de la clase Topo del framework de mininet de Python. Para personalizar la red, se deben reemplazar las funciones correspondientes de la clase base.

`--topo custom,[switch_amount]`: permite configurar la topología de forma personalizada. Existen otras opciones para el comando `--topo` *(linear, single…)*, pero en nuestro caso debe ser custom para poder personalizar la red a nuestra manera. El segundo parámetro es la cantidad de switches que tendrá la red. Dando como resultado una topología preestablecida de forma que los switches formen una cadena, similar a la imagen presentada a continuación:

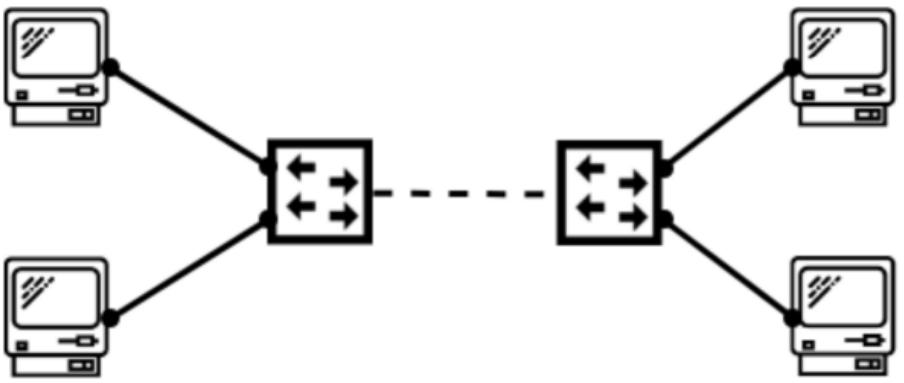


Figura 2

`--mac`: permite que las direcciones IP y MAC de las subred sean pequeñas, únicas y fáciles de leer.

`--controller remote`: permite configurar las reglas de los switches a partir de una herramienta externa (en nuestro caso, POX, como se explica en el siguiente apartado).

Sobre la configuración de reglas

Cómo se definen las reglas de ruteo

La ejecución del comando se hace con los siguientes argumentos:

```
./pox.py misc.firewall forwarding.l2_learning
```

`misc.firewall`: ruta del archivo en el cual se define la clase que establecerá las reglas de ruteo. En nuestro caso, es la clase *Firewall*, que hereda de la clase *EventMixin*, propia del framework *pox* de Python. Para configurar reglas de ruteo personalizadas, se deben sobreescribir algunos métodos propios de la clase base.

`forwarding.l2_learning`: genera que los switches actúen como switches del tipo *l2_learning*. Un *l2 switch* (layer-2 switch), es un switch a nivel de capa de enlace, que utiliza direcciones MAC para determinar la ruta a través de la cual se transportan los frames. Para poder lograrlo, estos switches deben aprender dónde están localizados los otros switches y utilizar flujos reactivos para manejar el tráfico a partir de las direcciones MAC mencionadas.

Sobre las reglas de ruteo

Se define un archivo `settings.py` que permite definir y variar los parámetros de cada una de las 3 reglas por defecto designadas por la consigna. Dentro de este se podrá, por ejemplo respecto de la regla 1, variar el puerto destino bloqueado; respecto de la regla 2, el host emisor, puerto destino y protocolo; etc.
Aún así, estas están predeterminadas según indica la consigna.

Las reglas son generadas por la clase `GenericBlockRule` que permite generar tanto las 3 reglas por defecto como cualquier combinación de IPs, puertos y protocolos, solo basta con generar en el archivo de configuraciones una configuración del firewall deseado y proveer al array de reglas activas `ACTIVE_RULES`.

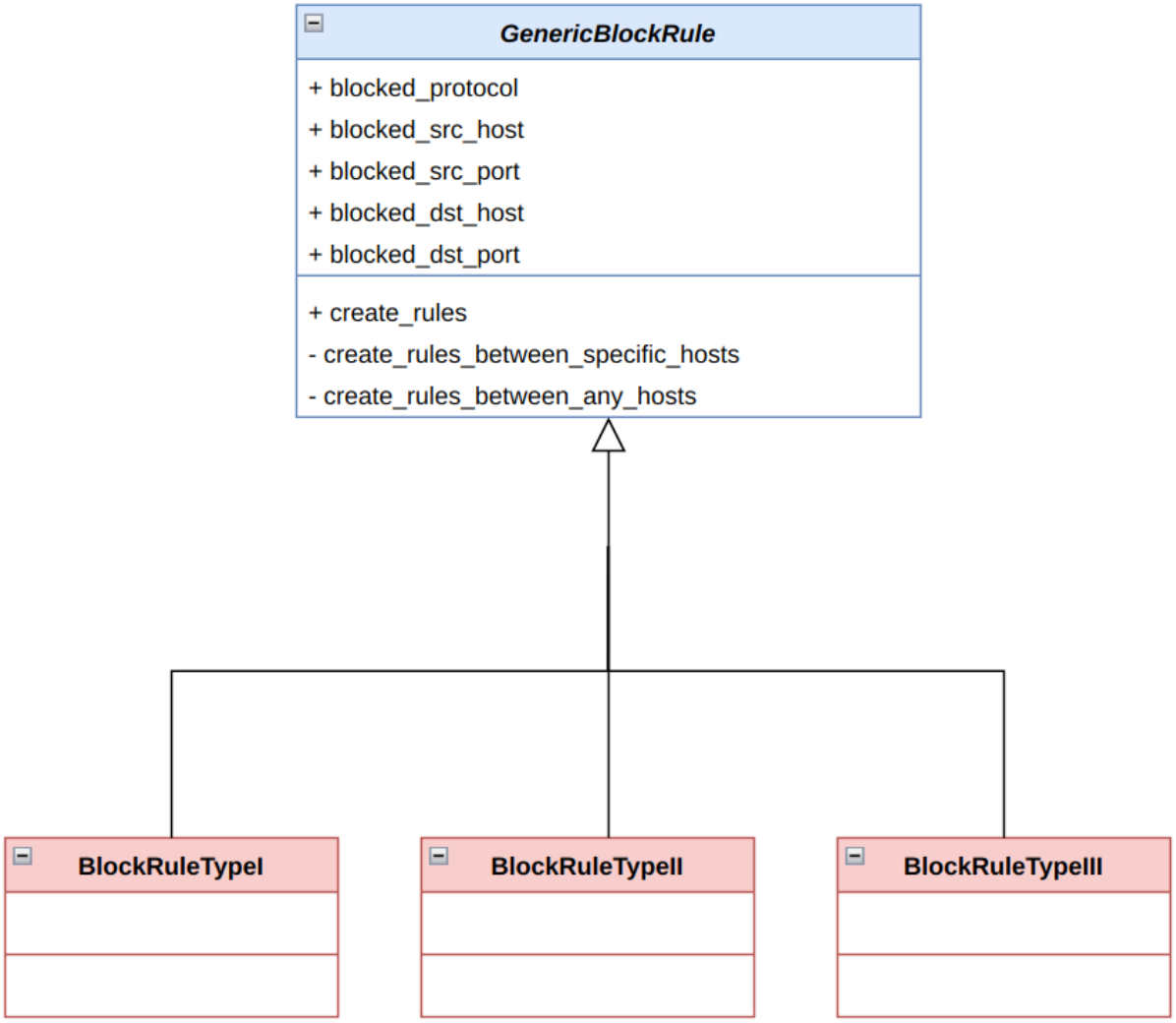


Figura 3. Diagrama de clases

Las claves que deberá tener una regla se deben corresponder con las provenientes de la documentación de POX, de las cuales se destacan:

`ofp_match` attributes:

Attribute	Meaning
in_port	Switch port number the packet arrived on
dl_src	Ethernet source address
dl_dst	Ethernet destination address
dl_vlan	VLAN ID
dl_vlan_pcp	VLAN priority
dl_type	Ethertype / length (e.g. 0x0800 = IPv4)
nw_tos	IP TOS/DS bits
nw_proto	IP protocol (e.g., 6 = TCP) or lower 8 bits of ARP opcode
nw_src	IP source address
nw_dst	IP destination address
tp_src	TCP/UDP source port
tp_dst	TCP/UDP destination port

Figura 4. Estructura del match

Un único switch como Firewall

Por enunciado se sabe que las reglas de ruteo deben ser configurables, sin embargo, la cantidad de switches que aplican estas reglas está fijado en uno. Esto implica, que solamente uno de los N switches de la red actúa como Firewall.

Esto implica necesariamente lo siguiente:

- Si el switch controlador está en uno de los extremos de la red, entonces la comunicación de los hosts del otro extremo estará libre de la aplicación de reglas.
- Si el switch controlador no está en ninguno de los extremos de la red, entonces la comunicación entre los hosts de ambos extremos estará libre de reglas.
- Cualquiera sea el switch que actúe como Firewall, la comunicación entre los hosts de un extremo y otro se verá afectada por las reglas establecidas.

Reglas de bloqueo

Los paquetes pueden bloquearse en base a cinco campos distintos:

Figura 5. Campos de interés

Cualquier combinación de estos cinco campos es válida, a excepción de las siguientes:

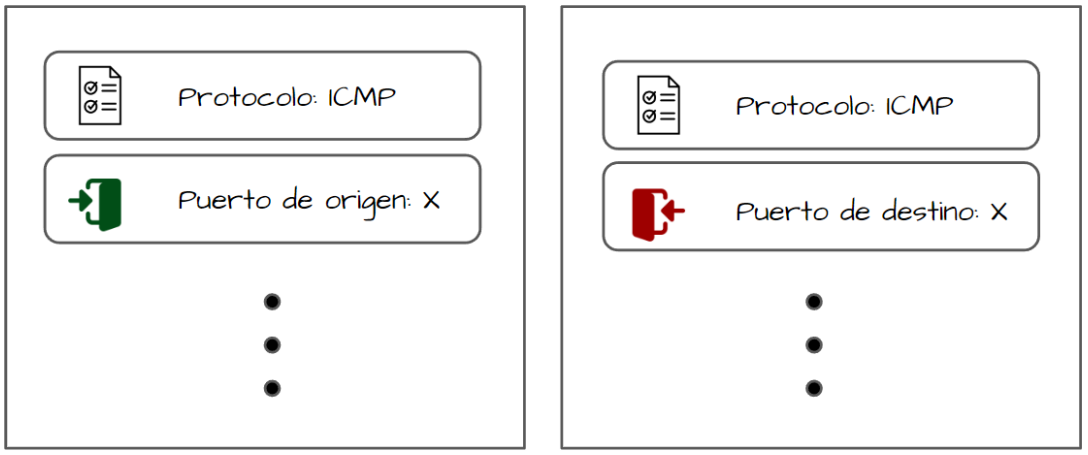


Figura 6. Combinaciones inválidas.

- Se especifica ICMP como protocolo a bloquear, y se especifica alguno de los dos puertos (origen o destino). Como ICMP no utiliza puertos, esta regla no tiene sentido, por lo que se lanza una excepción.

Se lanzará una **advertencia** si se especifica una regla que cumpla con lo siguiente:

- Se especifica un puerto de origen a bloquear y TCP como protocolo a bloquear. Si bien esta regla es válida, la misma no puede testearse con *iperf* (no permite especificar un puerto de origen) ni con *iperf3* (ejecuta un handshake TCP antes de empezar a enviar mensajes, sin importar el protocolo elegido).

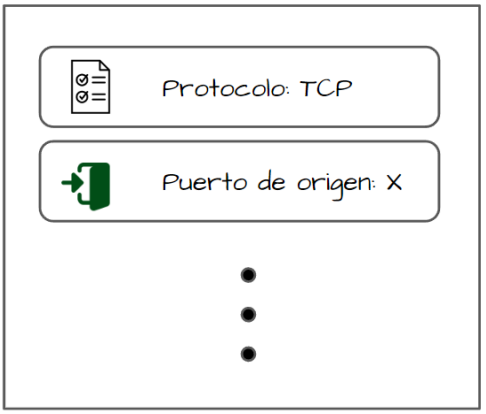


Figura 7. Regla basada en puerto origen.

Hipótesis sobre reglas de bloqueo

En base al enunciado del trabajo, se identificaron dos tipos de reglas que actúan de forma distinta y no tienen relación con las reglas por defecto de la consigna:

1. Si se especifican ambos hosts en la regla, la misma actúa específicamente sobre la comunicación entre esos dos hosts **(Regla A)**
2. Si alguno de los hosts no es especificado, la regla actúa de forma más general, y la comunicación entre varios hosts puede ser alcanzada por la misma **(Regla B)**.

Para la **Regla A,** se define el siguiente funcionamiento:

- Si solamente se especifican dos hosts, entonces *los hosts no pueden comunicarse entre ellos de ninguna manera* y en ninguna dirección, según especificación del enunciado.
- Si además de definirse ambos hosts, se definen ambos puertos, entonces solamente esos puertos de dichos hosts están incomunicados entre sí. Por ejemplo, si la regla de bloqueo es la siguiente:

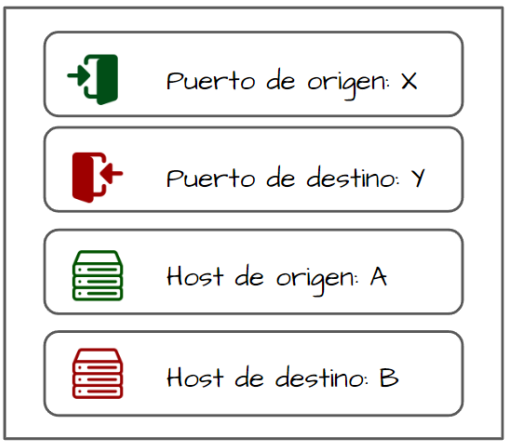


Figura 8

En este caso, lo que sucederá es que el puerto X de A no podrá comunicarse con el puerto Y de B. El puerto X de A podrá comunicarse libremente con cualquier otro puerto de B, y lo análogo para el puerto Y de B.

- Si se define un protocolo a bloquear, el bloqueo funcionará de forma análoga a los puntos anteriores, pero solamente si el protocolo utilizado es el especificado.

El funcionamiento de la **Regla B** puede deducirse directamente del enunciado, y no da lugar a diferentes interpretaciones: se bloqueará cualquier paquete que cumpla con los campos especificados.

Pruebas

Para las pruebas se decidió utilizar la topología definida previamente en el apartado Sobre la topología de la red.

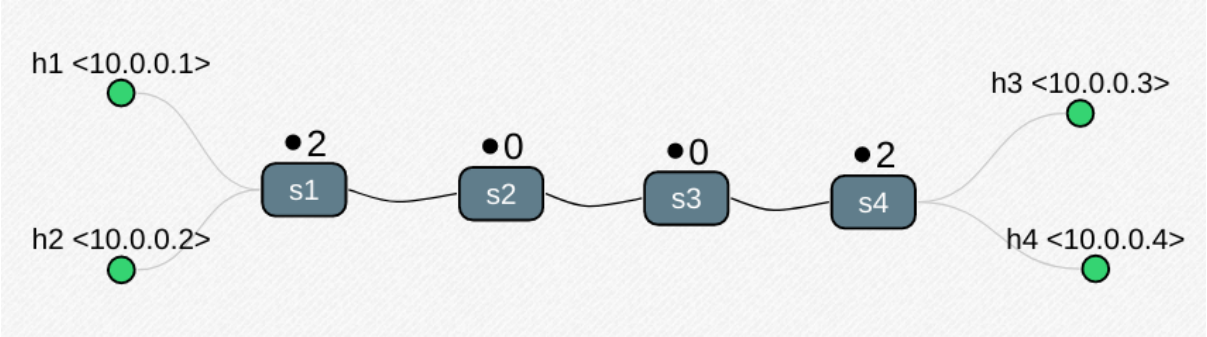


Figura 9

En todos los casos, a menos que se mencione explícitamente, el firewall definido con las reglas se ubica en el switch s1. Esto significa que dependiendo el caso y la dirección del flujo, se podrán o no observar paquetes OpenFlow en Wireshark.

Regla 1

Se deben descartar todos los mensajes cuyo puerto destino sea 80.

Wireshark

En ambos casos se ubica el Firewall en el switch s4.

Caso 1:

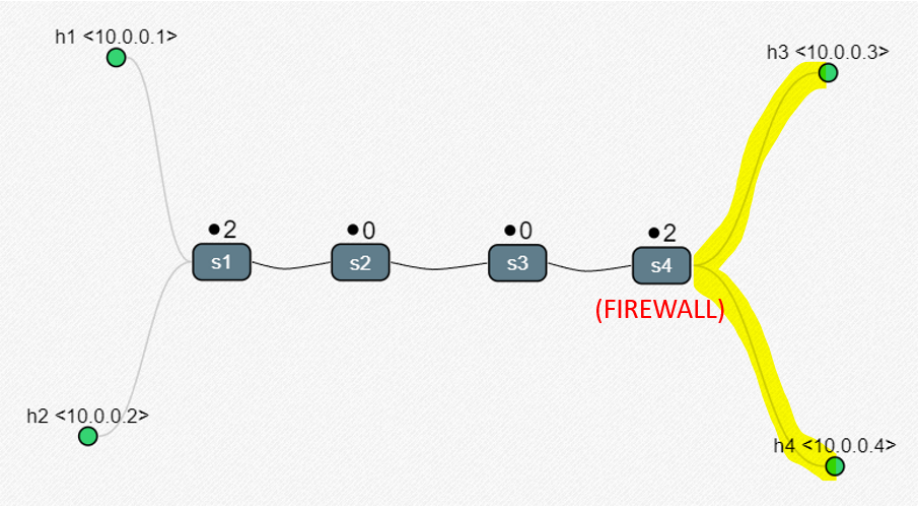


Figura 10. Regla 1. Caso 1

Realizado con servidor seteado en el Host 4 (10.0.0.4) y el cliente en el Host 3 (10.0.0.3). Filtro de Wireshark:

```
(ip.src == 10.0.0.3 || ip.src == 10.0.0.4)
```

Enviando por protocolo UDP, desde el h3 al h4, se puede observar que solo se envían paquetes UDP y no se evidencian paquetes OpenFlow dado que es el primer switch por el que cada paquete pasa y son filtrados por el firewall.

No.	Time	Source	Destination	Protocol	Length	Info
21003	61.561740270	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21009	61.573053660	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21010	61.573061364	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21014	61.584277982	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21018	61.595497005	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21023	61.606712591	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21028	61.617927205	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21032	61.629146117	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21037	61.640360971	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21041	61.651570776	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21046	61.662785752	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21050	61.674002300	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21055	61.685224087	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21058	61.696442318	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21064	61.707647595	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21067	61.718870184	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21073	61.730085249	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21077	61.741298851	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21081	61.752523855	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21090	61.763751614	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21095	61.774944918	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21099	61.786159211	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21100	61.797372172	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21102	61.808589692	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21103	61.819807653	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21105	61.831027426	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21106	61.842236300	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21108	61.853446445	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21109	61.864663584	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21111	61.875880663	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21112	61.887041977	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21114	61.898309430	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21115	61.909527131	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21118	61.920741605	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21123	61.931936943	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21127	61.943171795	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21130	61.954401477	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21137	61.965603167	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21144	61.976807993	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21151	61.988034309	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21156	61.999242381	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21162	62.010462515	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21169	62.021672431	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21177	62.032891894	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21184	62.044109023	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21187	62.055324349	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21192	62.066534414	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470
21197	62.077753637	10.0.0.4	10.0.0.3	UDP	1514	37799 → 80 Len=1470

Captura 1

Caso 2:

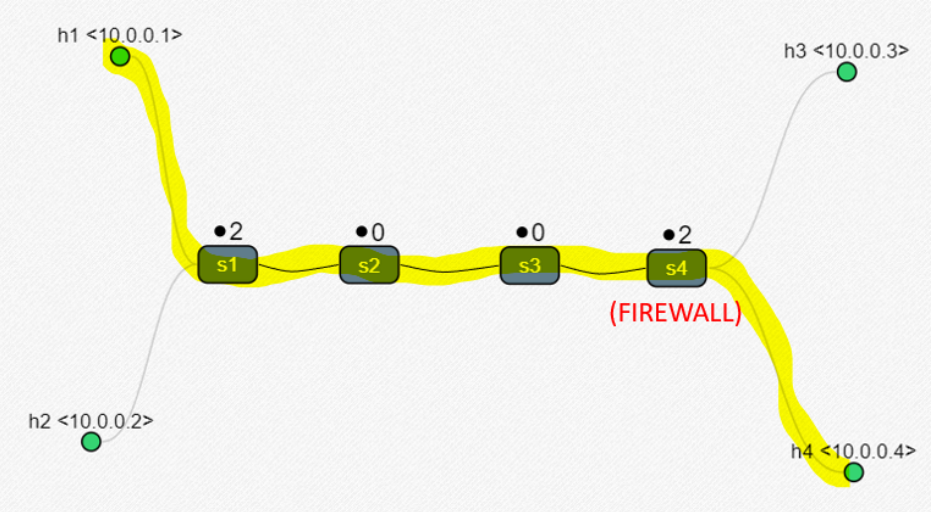


Figura 11. Regla 1. Caso 2

Cambiando de protocolo a TCP y enviando desde el h1 al h4, se evidencia que los paquetes OpenFlow viajarán por los switches hasta llegar al firewall donde serán bloqueados.

No.	Time	Source	Destination	Protocol	Length	Info
2496	7.846778457	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
2498	7.848313112	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
2502	7.848501003	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
2504	7.851046851	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
2509	7.851197193	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
2513	7.853820546	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
12677	39.194799441	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
12687	39.220212887	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
12692	39.220556130	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
12694	39.223139328	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
12699	39.223390859	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
12700	39.225946094	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
25487	73.238705893	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
25489	73.241368830	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
25494	73.241638014	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
25504	73.242987631	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT
25518	73.243206030	10.0.0.1	10.0.0.4	OpenFlow	160	Type: OFPT_PACKET_IN
25522	73.244617303	10.0.0.1	10.0.0.4	OpenFlow	254	Type: OFPT_PACKET_OUT

Captura 2

Logs Controlador

Caso 1: Enviando por protocolo UDP, desde el h3 al h4, no hay logs.

Caso 2: Si ahora se envía por protocolo TCP desde el h1 al h4, la información pasará por varios switches (en este caso 3) hasta llegar al s4 donde se encuentra el firewall.

```
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: TCP Protocol
INFO:misc.firewall: PORT SRC: 51046
INFO:misc.firewall: PORT DST: 80
INFO:misc.firewall: ##########################
```

Este log se repite 9 veces, 3 por switch.

Iperf

Caso 1: Al tener ubicado el firewall en el switch s4 y enviando por protocolo UDP, desde el h3 al h4, se puede observar que efectivamente se bloquean los paquetes ya que no se reciben los ACK.

```
Intro/intro-distribuidos-TP2# iperf -c 10.0.0.3 -p 80 -u
------------------------------------------------------------
Client connecting to 10.0.0.3, 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 37799 connected with 10.0.0.3 port 80
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0153 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  5] WARNING: did not receive ack of last datagram after 10 tries.
```

Captura 3

Caso 2: Si ahora se envía por protocolo TCP desde el h1 al h4, la información pasará por varios switches hasta llegar al s4 donde se encuentra el firewall.
De esta forma, iperf da como resultado un timeout.

```
Intro/intro-distribuidos-TP2# iperf -c 10.0.0.4 -p 80
tcp connect failed: Connection timed out
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 80
TCP window size: -1.00 Byte (default)
------------------------------------------------------------
[  1] local 0.0.0.0 port 0 connected with 10.0.0.4 port 80
```

Captura 4

Regla 2

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

Wireshark

Caso 1:

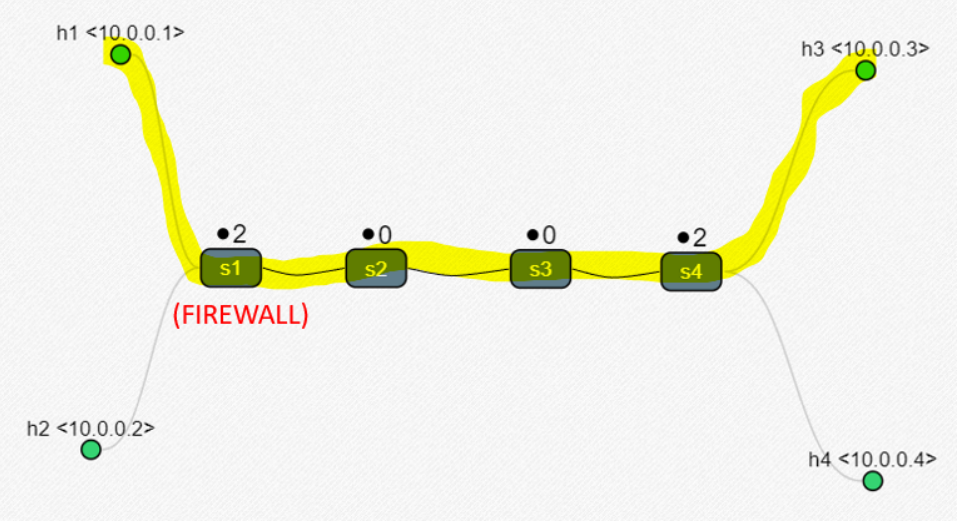


Figura 12. Regla 2. Caso 1

Ejemplo realizado con servidor seteado en el Host 3 (10.0.0.3) y el cliente en el Host 1 (10.0.0.1). Filtro de Wireshark:

```
(ip.src == 10.0.0.3 || ip.src == 10.0.0.1)
```

En este caso, se pueden observar muchos paquetes UDP que nunca llegan a pasar del primer switch (esto es porque **el firewall está en el s1**).

No.	Time	Source	Destination	Protocol	Length	Info
460	2.115594812	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
462	2.126896125	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
463	2.126902697	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
464	2.138114623	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
468	2.149344333	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
469	2.160558923	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
472	2.171792239	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
474	2.182996531	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
479	2.194195192	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
480	2.205401598	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
482	2.216617100	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
486	2.227830259	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
495	2.239058475	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
500	2.250245895	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
502	2.261489060	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
505	2.272722686	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
509	2.283941916	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
512	2.295155064	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
516	2.306351671	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
519	2.317573446	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
524	2.328772117	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
527	2.339987278	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
533	2.351201539	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
538	2.362423153	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
543	2.373641491	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
547	2.384845342	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
551	2.396083606	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
558	2.407308357	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
562	2.418506457	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
566	2.429714074	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
572	2.440938814	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
579	2.452173844	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
582	2.463376762	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
585	2.474575193	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
588	2.485790585	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
594	2.497012270	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
598	2.508222522	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
602	2.519440870	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
608	2.530651394	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
613	2.541874861	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
616	2.553088129	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
622	2.564310174	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
629	2.575524064	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
632	2.586718868	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
636	2.597954618	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
640	2.609182283	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
648	2.620392095	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470
652	2.631606766	10.0.0.1	10.0.0.3	UDP	1514	36470 → 5001 Len=1470

Captura 5

Caso 2:

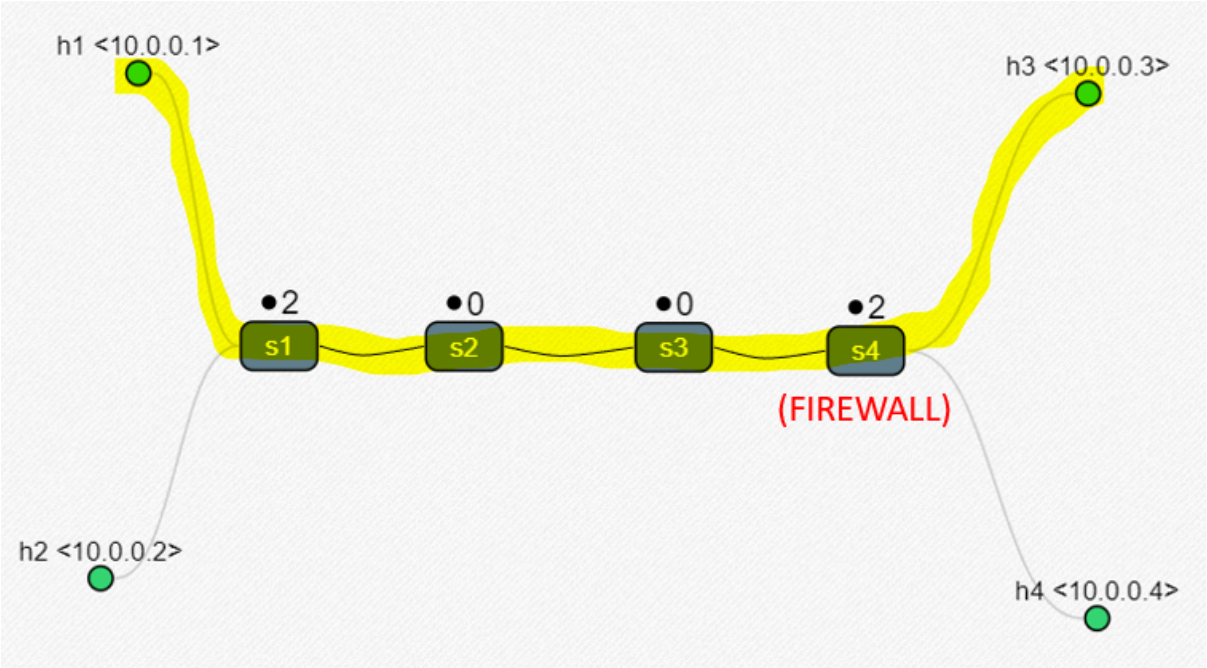


Figura 13. Regla 2. Caso 2

Ahora bien, **si se reubica el firewall al switch s4**, se podrá observar el flujo de paquetes OpenFlow atravesando los 3 switches intermedios (s1, s2, s3) hasta llegar al cuarto donde se filtra el flujo.

No.	Time	Source	Destination	Protocol	Length	Info
3906	13.391370619	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3907	13.391572879	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3912	13.402663189	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3913	13.402672817	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3914	13.402755112	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3916	13.402772033	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3920	13.404951343	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3922	13.405025863	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3923	13.405029019	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3924	13.405029850	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3925	13.405166457	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3928	13.406496673	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3930	13.406546827	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3931	13.406548971	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3932	13.406549432	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3933	13.406603293	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3935	13.408013960	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3937	13.408068873	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3938	13.408069915	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3939	13.408173600	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3941	13.408338770	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3943	13.408370279	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3944	13.408372212	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3945	13.408372623	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3946	13.408426133	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3947	13.409864432	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3949	13.409935295	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3950	13.409936408	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3951	13.410009475	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3954	13.410248152	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3956	13.410301723	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3957	13.410303326	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3959	13.411640385	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT
3961	13.411700718	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3962	13.411702321	10.0.0.1	10.0.0.3	UDP	1514	57765 → 5001 Len=1470
3963	13.411765870	10.0.0.1	10.0.0.3	OpenFlow	1598	Type: OFPT_PACKET_IN
3964	13.411905923	10.0.0.1	10.0.0.3	OpenFlow	1604	Type: OFPT_PACKET_OUT

Captura 6

Logs Controlador

Caso 1:
No habrá información en los logs ya que el firewall en s1 bloquea los paquetes generando que no puedan continuar con su recorrido.

Caso 2:
Ubicando al firewall en el switch s4:

```
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: UDP Protocol
INFO:misc.firewall:  PORT SRC: 57765
INFO:misc.firewall:  DST PORT: 5001
INFO:misc.firewall: ##########################
```

Esta información se repetirá muchas veces dado que los paquetes atravesarán varios switches.

Iperf

Caso 1:
Firewall ubicado en switch s1. No se reciben los ACK.

```
Intro/intro-distribuidos-TP2# iperf -c 10.0.0.3 -u
------------------------------------------------------------
Client connecting to 10.0.0.3, 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 33260 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0153 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  5] WARNING: did not receive ack of last datagram after 10 tries.
```

Captura 7

Caso 2:
Firewall ubicado en switch s4. No reciben los ACK.

```
Intro/intro-distribuidos-TP2# iperf -c 10.0.0.3 -u
------------------------------------------------------------
Client connecting to 10.0.0.3, 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 57765 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0153 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  5] WARNING: did not receive ack of last datagram after 10 tries.
```

Captura 8

Vemos que en ambos casos, por más que el firewall esté ubicado en distintos switches los paquetes resultan bloqueados indefectiblemente.

Regla 3

Se debe elegir dos hosts cualquiera, y los mismos no deben poder comunicarse de ninguna forma.

Wireshark

Caso 1:

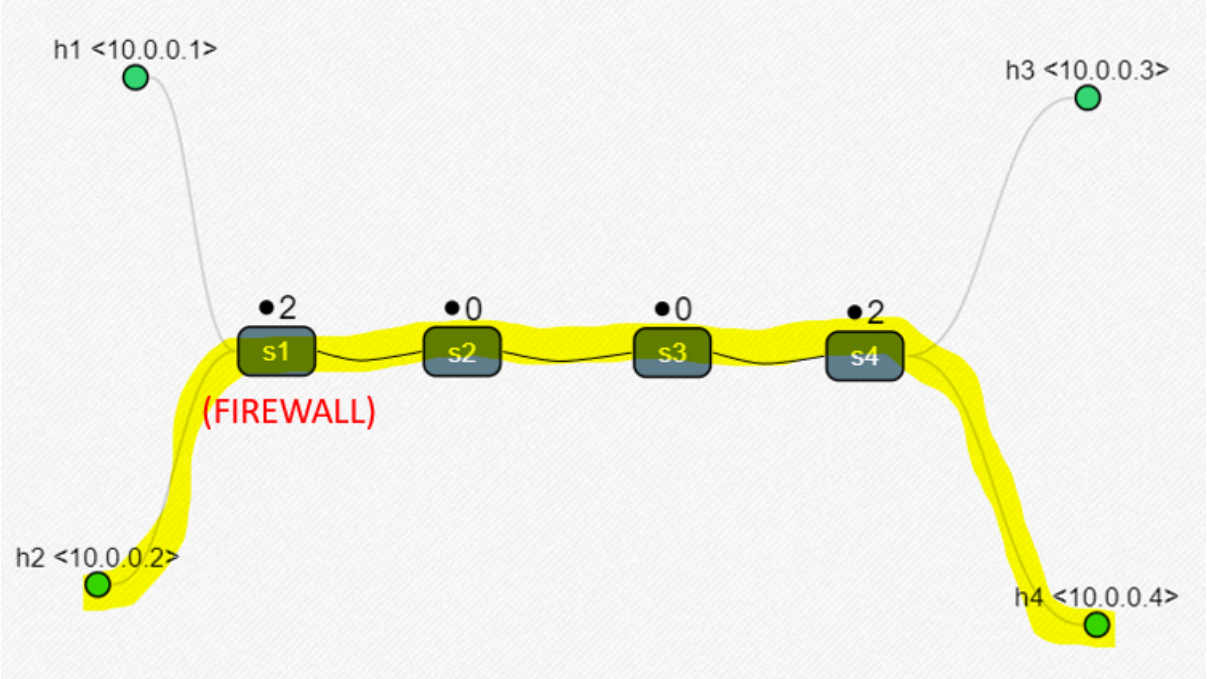
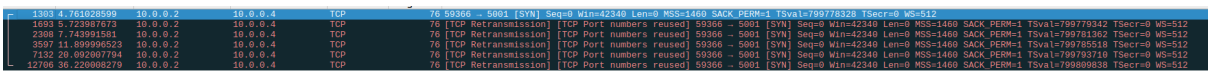


Figura 14. Regla 3. Caso 1 y caso 2

Ejemplo realizado con servidor seteado en el Host 4 (10.0.0.4) y el cliente en el Host 2 (10.0.0.2), es decir **de h2 a h4**. Recordemos que el firewall se encuentra en el s1.
Filtro de Wireshark:

```
(ip.src == 10.0.0.2 || ip.src == 10.0.0.4)
```

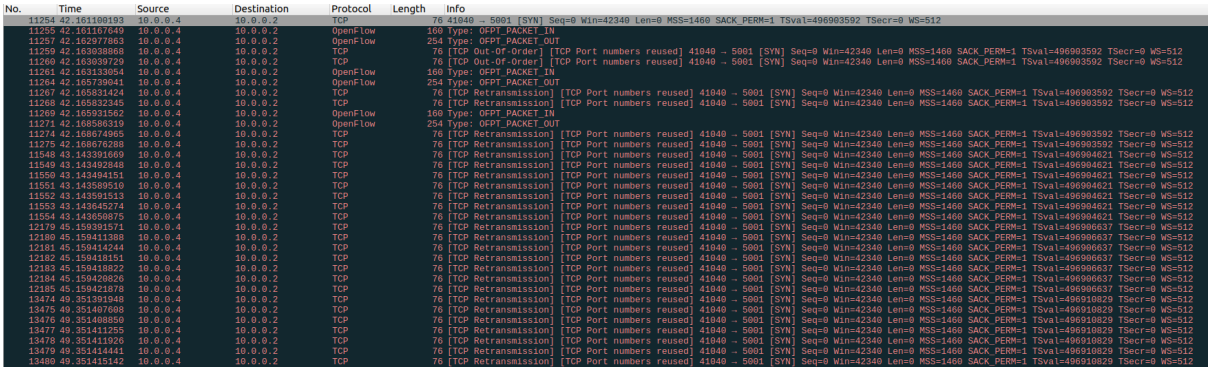


Captura 9

Dado que los paquetes son bloqueados en el primer switch por el que pasan, no se observan paquetes OpenFlow y sólamente se observan los fallidos del protocolo TCP utilizado al realizar *iperf*.

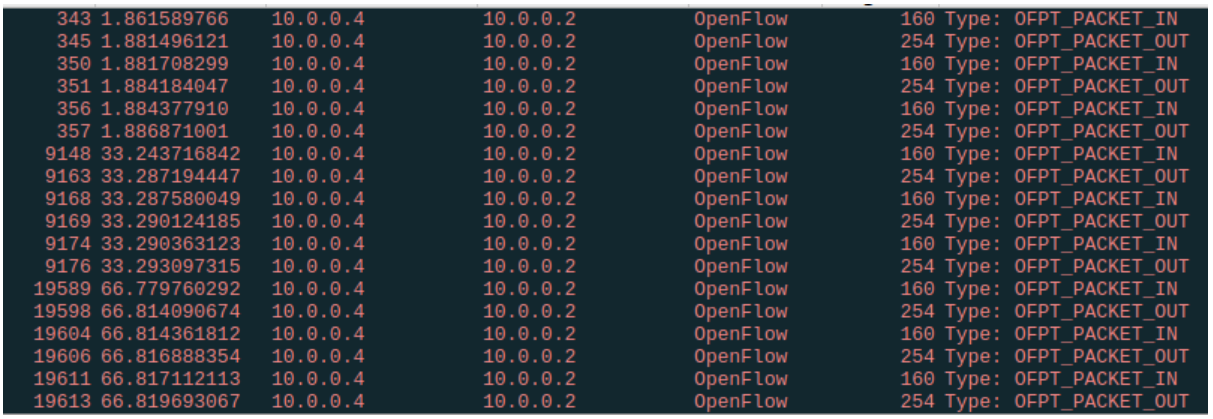
Caso 2:

En este caso se realiza el camino inverso, **de h4 a h2**, de forma que se puedan observar los paquetes en viaje dado que el firewall se encuentra en el s1.



No.	Time	Source	Destination	Protocol	Length	Info
11254	42.161100193	10.0.0.4	10.0.0.2	TCP		76 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11255	42.161167649	10.0.0.4	10.0.0.2	OpenFlow		160 Type: OFPT_PACKET_IN
11257	42.162977863	10.0.0.4	10.0.0.2	OpenFlow		254 Type: OFPT_PACKET_OUT
11259	42.163038868	10.0.0.4	10.0.0.2	TCP		76 [TCP Out-Of-Order] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11260	42.163039729	10.0.0.4	10.0.0.2	TCP		76 [TCP Out-Of-Order] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11261	42.163133054	10.0.0.4	10.0.0.2	OpenFlow		160 Type: OFPT_PACKET_IN
11264	42.165739041	10.0.0.4	10.0.0.2	OpenFlow		254 Type: OFPT_PACKET_OUT
11267	42.165831424	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11268	42.165832345	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11269	42.165931562	10.0.0.4	10.0.0.2	OpenFlow		160 Type: OFPT_PACKET_IN
11271	42.168586319	10.0.0.4	10.0.0.2	OpenFlow		254 Type: OFPT_PACKET_OUT
11274	42.168674965	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11275	42.168676288	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496903592 TSecr=0 WS=512
11548	43.143391669	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11549	43.143492848	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11550	43.143494151	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11551	43.143589510	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11552	43.143591513	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11553	43.143645274	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
11554	43.143650875	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496904621 TSecr=0 WS=512
12179	45.159391571	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12180	45.159411308	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12181	45.159414244	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12182	45.159418151	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12183	45.159418822	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12184	45.159420826	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
12185	45.159421878	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496906637 TSecr=0 WS=512
13474	49.351391948	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13475	49.351407698	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13476	49.351408850	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13477	49.351411255	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13478	49.351411926	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13479	49.351414441	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512
13480	49.351415142	10.0.0.4	10.0.0.2	TCP		76 [TCP Retransmission] [TCP Port numbers reused] 41040 → 5001 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=496910829 TSecr=0 WS=512

Captura 10



343	1.861589766	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
345	1.881496121	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
350	1.881708299	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
351	1.884184047	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
356	1.884377910	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
357	1.886871001	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
9148	33.243716842	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
9163	33.287194447	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
9168	33.287580049	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
9169	33.290124185	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
9174	33.290363123	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
9176	33.293097315	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
19589	66.779760292	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
19598	66.814090674	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
19604	66.814361812	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
19606	66.816888354	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT
19611	66.817112113	10.0.0.4	10.0.0.2	OpenFlow	160 Type: OFPT_PACKET_IN
19613	66.819693067	10.0.0.4	10.0.0.2	OpenFlow	254 Type: OFPT_PACKET_OUT

Captura 11

Así, se evidencia que hay paquetes fallidos de protocolo TCP y los paquetes OpenFlow correspondientes a los switches por donde hacen su recorrido.

Logs Controlador

Caso 1: No se evidencian logs.

Caso 2: Los paquetes viajan por varios switches hasta llegar al s1 donde se encuentra el firewall.

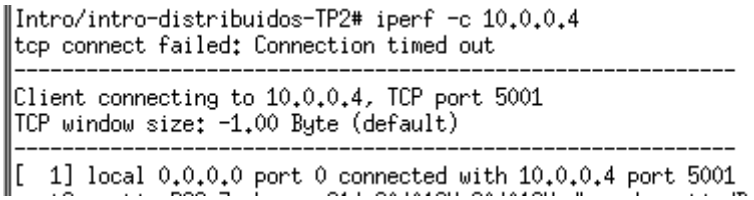
```
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: ##########################
```

```
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: #########################
INFO:misc.firewall:  IP SRC: 10.0.0.4
INFO:misc.firewall:  IP DST: 10.0.0.2
INFO:misc.firewall:  TCP Protocol
```

```
INFO:misc.firewall:  PORT SRC: 53702
INFO:misc.firewall:  PORT DST: 5001
INFO:misc.firewall: ##########################
```

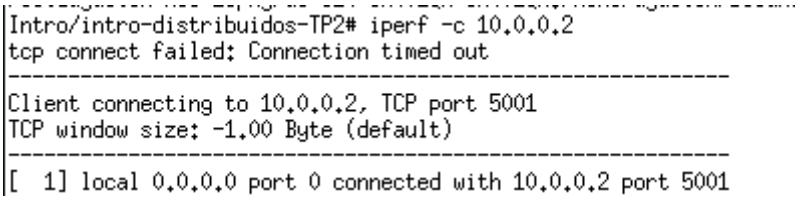
Iperf

Caso 1: Se puede observar que se genera un timeout en la conexión.



Captura 12

Caso 2: Se ve que finalmente se genera un timeout en la conexión.



Captura 13

En ambos casos, al utilizar protocolo TCP, se genera un timeout en la conexión.

Pingall

```
(ip.src == 10.0.0.1 || ip.src == 10.0.0.2 || ip.src == 10.0.0.3 ||
ip.src == 10.0.0.4 )
```

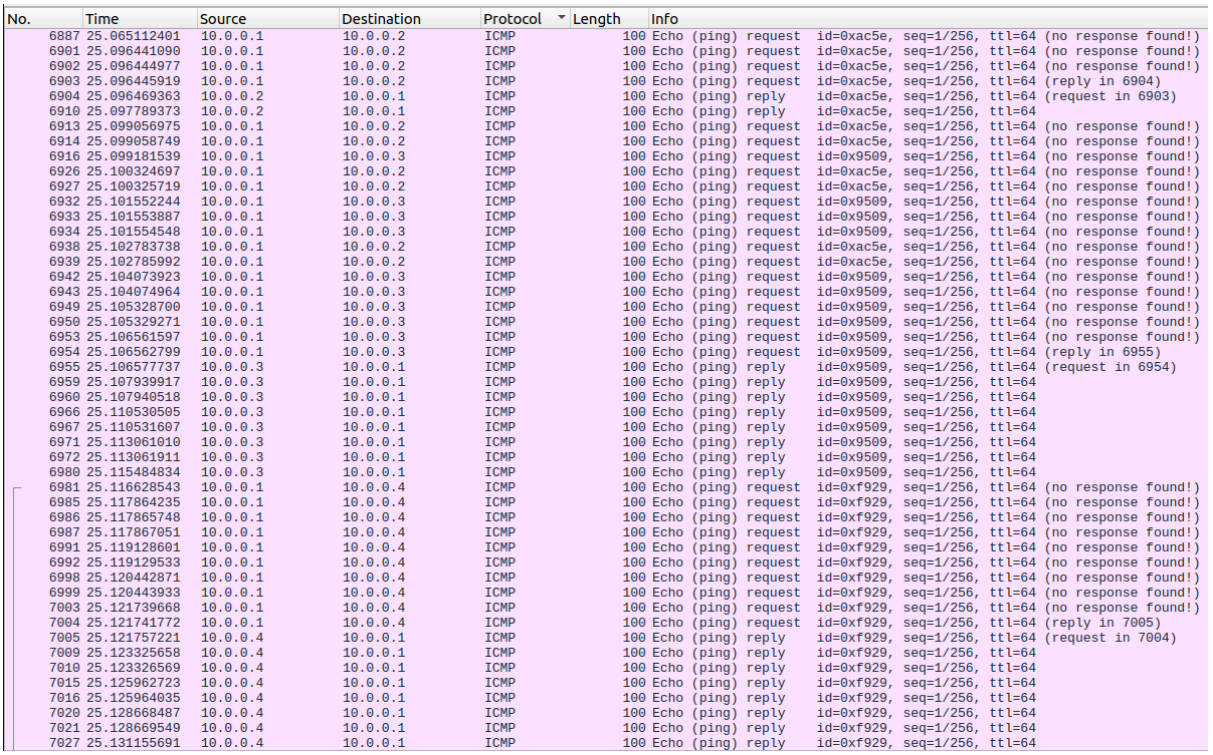
En este caso, se ejecuta el comando *pingall* para evidenciar la imposibilidad de los hosts (h2 y h4) de comunicarse.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 X
h3 -> h1 h2 h4
h4 -> h1 X h3
*** Results: 16% dropped (10/12 received)
```

Las siguientes capturas evidencian el envío de paquetes tanto OpenFlow como ICMP para realizar la ejecución del comando. Se puede observar que habrá paquetes OpenFlow que muestran el flujo entre los switches y además, paquetes ICMP pertinentes al ping entre los cuales hay requests (algunos que no reciben respuesta, por ser justamente la comunicación entre h2 y h4) y replies.

No.	Time	Source	Destination	Protocol	Length	Info
14743	45.289248996	10.0.0.3	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
14740	45.287979341	10.0.0.3	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
14735	45.287538212	10.0.0.4	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
14723	45.250696840	10.0.0.4	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10663	35.255626597	10.0.0.4	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10661	35.253177315	10.0.0.4	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10657	35.253037502	10.0.0.4	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10652	35.250631402	10.0.0.4	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10648	35.250515314	10.0.0.4	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10646	35.249291404	10.0.0.4	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10642	35.248018352	10.0.0.1	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
10639	35.245559862	10.0.0.1	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
10635	35.245390163	10.0.0.1	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
10633	35.242776913	10.0.0.1	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
10629	35.242595262	10.0.0.1	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
10626	35.240045611	10.0.0.1	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
10622	35.239942457	10.0.0.1	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
10620	35.238676489	10.0.0.1	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
10616	35.238517199	10.0.0.4	10.0.0.1	OpenFlow	278	Type: OFPT_PACKET_OUT
10613	35.236039774	10.0.0.4	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
10609	35.235899912	10.0.0.4	10.0.0.1	OpenFlow	278	Type: OFPT_PACKET_OUT
10607	35.233441062	10.0.0.4	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
10603	35.233283516	10.0.0.4	10.0.0.1	OpenFlow	278	Type: OFPT_PACKET_OUT
10599	35.230820267	10.0.0.4	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
10595	35.230689571	10.0.0.4	10.0.0.1	OpenFlow	278	Type: OFPT_PACKET_OUT
10593	35.229375412	10.0.0.4	10.0.0.1	OpenFlow	184	Type: OFPT_PACKET_IN
10589	35.228130222	10.0.0.4	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10587	35.226873080	10.0.0.4	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10583	35.226710194	10.0.0.3	10.0.0.4	OpenFlow	278	Type: OFPT_PACKET_OUT
10579	35.225511502	10.0.0.3	10.0.0.4	OpenFlow	184	Type: OFPT_PACKET_IN
10574	35.224066126	10.0.0.2	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10572	35.221590174	10.0.0.2	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10568	35.221423100	10.0.0.2	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10564	35.219105235	10.0.0.2	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10559	35.218902804	10.0.0.2	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10556	35.216383971	10.0.0.2	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10551	35.216270669	10.0.0.2	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10548	35.215013426	10.0.0.2	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN
10543	35.214873914	10.0.0.3	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10541	35.212438939	10.0.0.3	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10536	35.212261706	10.0.0.3	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10532	35.209766337	10.0.0.3	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10528	35.209642294	10.0.0.3	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10526	35.207160802	10.0.0.3	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10521	35.207017031	10.0.0.3	10.0.0.2	OpenFlow	278	Type: OFPT_PACKET_OUT
10518	35.205734351	10.0.0.3	10.0.0.2	OpenFlow	184	Type: OFPT_PACKET_IN
10513	35.204495944	10.0.0.1	10.0.0.3	OpenFlow	278	Type: OFPT_PACKET_OUT
10511	35.202074484	10.0.0.1	10.0.0.3	OpenFlow	184	Type: OFPT_PACKET_IN

Captura 14



No.	Time	Source	Destination	Protocol	Length	Info
6887	25.065112401	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6901	25.096441090	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6902	25.096444977	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6903	25.096445919	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (reply in 6904)
6904	25.096469363	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0xac5e, seq=1/256, ttl=64 (request in 6903)
6910	25.097789373	10.0.0.2	10.0.0.1	ICMP	100	Echo (ping) reply id=0xac5e, seq=1/256, ttl=64
6913	25.099056975	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6914	25.099058749	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6916	25.099181539	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6926	25.100324697	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6927	25.100325719	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6932	25.101552244	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6933	25.101553887	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6934	25.101554548	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6938	25.102783738	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6939	25.102785992	10.0.0.1	10.0.0.2	ICMP	100	Echo (ping) request id=0xac5e, seq=1/256, ttl=64 (no response found!)
6942	25.104073923	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6943	25.104074964	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6949	25.105328700	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6950	25.105329271	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6953	25.106561597	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (no response found!)
6954	25.106562799	10.0.0.1	10.0.0.3	ICMP	100	Echo (ping) request id=0x9509, seq=1/256, ttl=64 (reply in 6955)
6955	25.106577737	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64 (request in 6954)
6959	25.107939917	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6960	25.107940518	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6966	25.110530505	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6967	25.110531607	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6971	25.113061010	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6972	25.113061911	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6980	25.115484834	10.0.0.3	10.0.0.1	ICMP	100	Echo (ping) reply id=0x9509, seq=1/256, ttl=64
6981	25.116628543	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6985	25.117864235	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6986	25.117865748	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6987	25.117867051	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6991	25.119128601	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6992	25.119129533	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6998	25.120442871	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
6999	25.120443933	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
7003	25.121739668	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (no response found!)
7004	25.121741772	10.0.0.1	10.0.0.4	ICMP	100	Echo (ping) request id=0xf929, seq=1/256, ttl=64 (reply in 7005)
7005	25.121757221	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64 (request in 7004)
7009	25.123325658	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7010	25.123326569	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7015	25.125962723	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7016	25.125964035	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7020	25.128668487	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7021	25.128669549	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64
7027	25.131155691	10.0.0.4	10.0.0.1	ICMP	100	Echo (ping) reply id=0xf929, seq=1/256, ttl=64

Captura 15

Logs

```
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: ##########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: ##########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: ##########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: ##########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: ##########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
```

```
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
```

```
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
```

```
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
```

```
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.2
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
```

```
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.1
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.1
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.2
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.4
INFO:misc.firewall: IP DST: 10.0.0.3
INFO:misc.firewall: ICMP Protocol
INFO:misc.firewall: #########################
INFO:misc.firewall: IP SRC: 10.0.0.3
INFO:misc.firewall: IP DST: 10.0.0.4
INFO:misc.firewall: ICMP Protocol
```

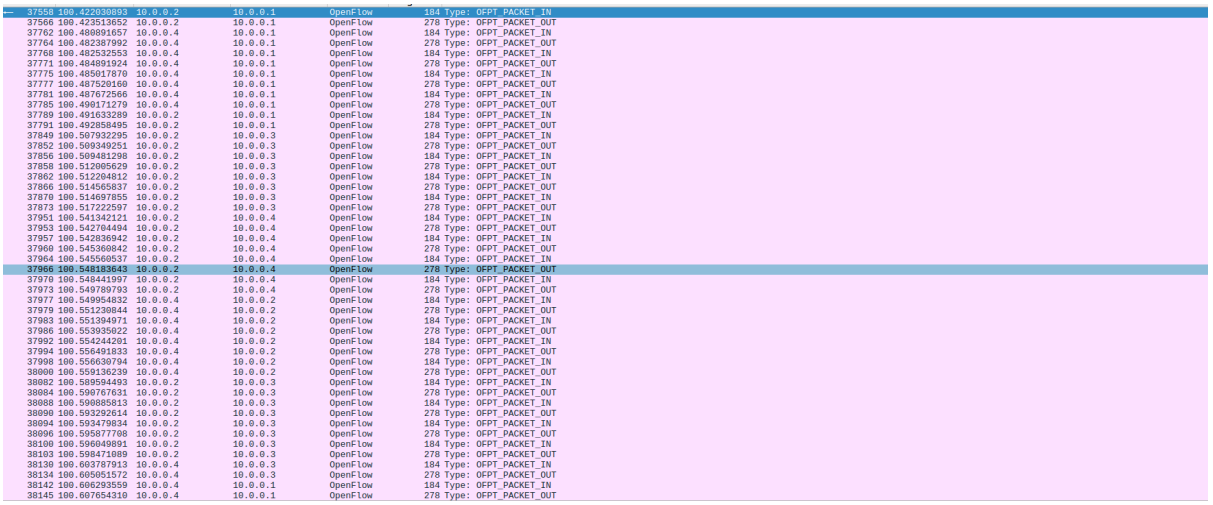
```
INFO:misc.firewall: ##########################
```

Este log abarca las combinaciones posibles de comunicaciones exceptuando que no pueden viajar mediante switches por la restricción de la regla entre h2 y h4 (notar que si hay logs entre h4 y h2 porque el firewall está en el switch 1 y pasa por 3 switches antes) en la ejecución de *pingall* en la consola de *mininet*.

Pingall

Aquí se muestra un ejemplo de la ejecución del comando *pingall* en la consola de *mininet*, en un momento en el que no existe ninguna regla aplicada.
Se puede observar que todos los paquetes llegan a destino correctamente.

```
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)
```



Captura 16

Preguntas

1. ¿Cuál es la diferencia entre un Switch y un router? ¿Qué tienen en común?

Tanto a los routers como a los link-layer switches se los denomina *packet switches*. Estos son dispositivos que cumplen la función de recibir paquetes por su canal de entrada, definir ,por algún criterio, a que canal de salida corresponde enviar cada paquete y llevar a cabo el envío por el canal correspondiente. A esta última acción se la llama forwarding.
La principal diferencia entre ambos radica en el criterio de decisión. Los routers realizarán forwarding basándose, en principio, en el direccionamiento IP mientras que los switches lo harán basándose en el direccionamiento MAC.

Ambos implementan las capas 1 y 2, física y enlace, mientras que el router implementa además la capa 3, de transporte. Esto significa que el switch no conoce el funcionamiento de direcciones IP para realizar el reenvío de paquetes.
Un router elige la ruta más corta para que un paquete llegue a destino. Un Switch almacena el paquete recibido, lo procesa para determinar su dirección de destino y lo reenvía a un destino específico.
La principal diferencia entre ambos es que un router conecta diferentes redes entre sí, mientras que un switch conecta varios dispositivos entre sí conformando así una red.

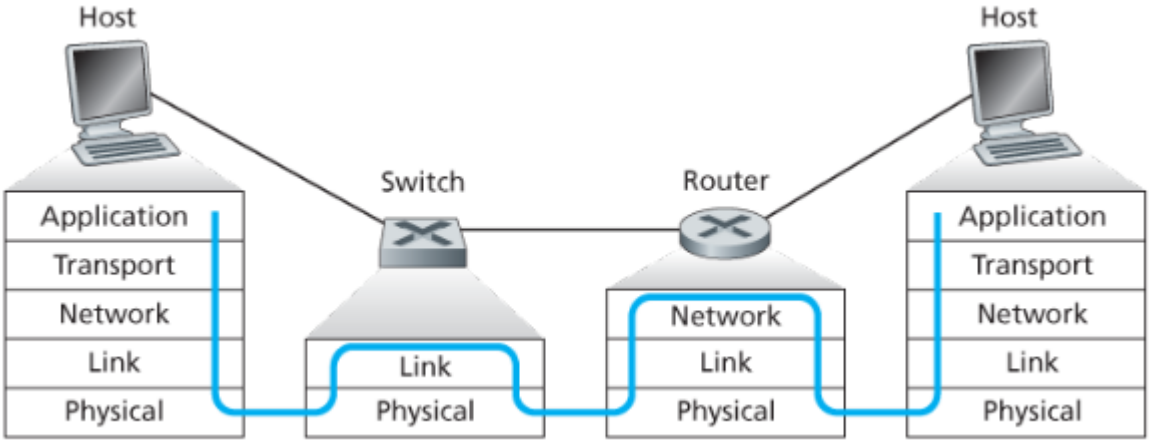


Figura 15. Dispositivos tradicionales.

De esta forma, un switch realiza su decisión de reenvío en base a la información del *frame* de la capa de enlace resultando en un dispositivo de capa 2 y un router realizar el reenvío en base a la información del *datagrama* del paquete de capa de red resultando en un dispositivo de capa 3.

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

Un switch convencional posee tanto el plano de datos como el plano de control, de forma que el ruteo y forwardeo se hace en el mismo dispositivo. En consecuencia, la modificación de la tabla y sus datos requiere de un operador o bien del procesador central que es es el encargado de ejecutar las funciones del plano de control, esto implica ejecutar los protocolos de routing, mantener las variables de estado y computar los valores de su tabla de forwarding.

Por otro lado, un switch OpenFlow implementa solamente el plano de datos mientras que el plano de control se implementa por fuera mediante software en un controlador específico que define las decisiones a tomar por parte del switch y se las comunica mediante paquetes del protocolo OpenFlow. Esto permite mayor flexibilidad para manejar el tráfico de paquetes y facilidad para establecer reglas y políticas de ruteo ya que el controlador tiene una visión más amplia de la red y puede permitirse ejecutar algoritmos más complejos de los que podría un switch convencional sin tener un alto impacto en el procesamiento (al tratarse de software puede tardar en el orden de los segundos).
Además, permite realizar balanceo de carga en la red de forma más eficiente.

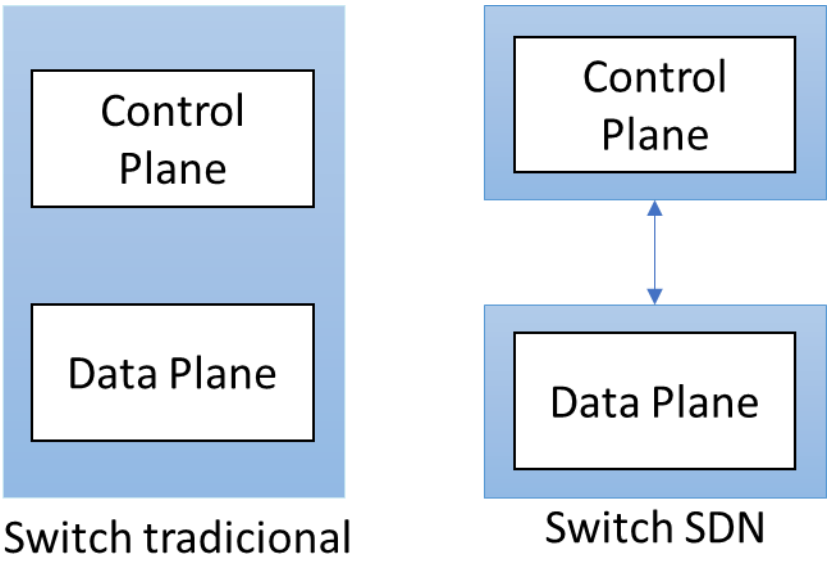


Figura 16. Arquitecturas de Routing.

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

Dado que la abstracción del match que hace OpenFlow permite que se tomen decisiones mediante software, en base a campos que pertenecen a los headers de los protocolos de

las capas 2, 3, y 4; estos dispositivos pueden funcionar como un router (capa 3) y como un switch (capa 2).
De esta forma un switch OpenFlow podría reemplazar la operativa que llevan a cabo los routers pero, dado que no se suelen tener en cuenta todos los campos del header IP, agregar este funcionamiento implicaría un impacto en la performance. Además, si se quisiese realizar ruteo entre ASes, esto implicaría que deba haber una centralización completa del controlador de OpenFlow para definir caminos para los paquetes y por ende habría necesariamente un gran costo de despliegue.
Por otro lado, los sistemas autónomos deberían disponer de independencia a la hora de decidir las acciones a tomar para sus routers. Si se contara con un único controlador central no se dispondría de tal independencia.

En conclusión, sería inviable tener un controlador central para toda la red de internet.

Conclusiones

En este trabajo, hemos utilizado herramientas que nos permiten:

- A través de la herramienta Mininet, pudimos establecer una red virtual, personalizando la cantidad de hosts y switches de la misma, así como los enlaces que los conectan.
- Modificar las tablas de ruteo de los switches de la red establecida (en particular, de un solo switch), para establecer reglas análogas a las establecidas en una SDN. Esto lo hicimos a través de la herramienta POX, que a su vez implementa el protocolo Openflow.
- Iperf e Iperf3 nos permitieron enviar mensajes entre los distintos hosts de la red virtual, dándonos la capacidad de establecer distintos parámetros a la hora de establecer la comunicación correspondiente.
- A partir de Wireshark, que ya había sido utilizada en el Trabajo Práctico 1 (File Transfer), pudimos hacer el seguimiento de los paquetes enviados en cada comunicación para verificar que las reglas de bloqueo efectivamente funcionaran de acuerdo a lo pretendido.

Las principales dificultades que encontramos en el desarrollo del trabajo radican en el proceso de aprendizaje de múltiples herramientas con las que hasta el momento no habíamos tenido ninguna experiencia, pero en cuanto pudimos entender de forma general cómo utilizarlas, avanzar con el desarrollo del código no fue tan complicado. El hecho de que la documentación de todas las herramientas fuera clara y concisa fue un factor que contribuyó a facilitar el trabajo. En este sentido, POX es la herramienta con mayor complejidad intrínseca, y la que más nos exigió a la hora de utilizarla correctamente.

Consideramos que las reglas soportadas por nuestro programa son relativamente flexibles, ya que permiten el bloqueo de paquetes a partir de cualquier combinación de los campos solicitados en el enunciado, y no se limitan solamente a adaptarse a los tres ejemplos específicos listados.

Bibliografía

Mininet Walkthrough

Official POX docs

OpenFlow Switch Specification Version 1.0.0