75.43 Introducción a los Sistemas Distribuidos

Trabajo Práctico N°2: Mininet (?)

Padrón	Alumno	Dirección de correo
104393	Arrachea, Tomás	tarrachea@fi.uba.ar
94194	Bravo Reyes, Christian	cbravor@fi.uba.ar
102896	Movia, Guido	gmovia@fi.uba.ar
99999	Pardo, Lucía	lpardo@fi.uba.ar
100815	Vazquez Lareu, Roman	rlareu@fi.uba.ar

Índice

1. Introducción

El presente trabajo práctico consiste en realizar una topología de red parametizable que poseera una cantidad de switches variables, formando uno cadena y en sus extremos tendrá dos hosts. Una vez realizada la misma, se procederá a probar su robustez mediante distintos ensayos, utilizando un controlador que permita aprender automaticamente a los switches, y funcione como un Firewall que deberá cumplir una serie de reglas.

2. Objetivo

El objetivo principal de este trabajo es la familiarización y comprensión de las distintas herramientas que nos permiten realizar las pruebas pertinentes sobre nuestra topología. Dentro de las mismas tenemos a Mininet, pox, Openflow y Wireshark.

3. Implementación

3.1. Topología

Como mencionamos brevemente en la introducción, la topología realizada consta de una cantidad de switches variables y en cada extremo posee dos hosts. La misma se encuentra implementada en el archivo **topo.py**. Para emular la misma vamos a utilizar **Mininet**.

Como primer paso, vamos a ejecutar la topología utilizando la siguiente instrucción a través de la consola: *sudo mn –custom topo.py –topo mytopo,<switches>–test pingall* . Siendo <switches>la cantidad de switches variables parametrizables. Al utilizar en la instrucción *–test pingall*, lo que se esta verificando es que ningún paquete se pierda en la red. Este es un test inicial que se realiza para analizar si la red esta lista para ser sometida pruebas más profundas utilizando distintos controladores.

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s0 s1 s2
*** Adding links:
(h1, s0) (h2, s0) (h3, s2) (h4, s2) (s1, s0) (s2, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s0 s1 s2 ...
*** Waiting for switches to connect
s0 s1 s2
*** 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)
*** Stopping 1 controllers
c0
*** Stopping 6 links
......
*** Stopping 3 switches
s0 s1 s2
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 7.431 seconds
```

Figura 1: Ejecución topología usando Mininet

Como podemos ver en la figura anterior, todos los paquetes fueron recibidos.

3.2. Controladores

A continuación, vamos a proceder a ejecutar sobre nuestra topología los controladores solicitados y analizar si hay pérdidas de paquetes, y qué sucede durante la ejecución.

3.2.1. L2 learning

Con este controlador, los switches OpenFlow se comportan como switches de aprendizaje de capa 2. De esta forma, el switch OpenFlow es capaz de reconocer la dirección MAC de los dispositivos conectados. Para aplicar este controlador sobre nuestra topología vamos a utilizar el metodo provisto por pox:

```
*** 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:
s0 s1 s2
*** Adding links:
(h1, s0) (h2, s0) (h3, s2) (h4, s2) (s1, s0) (s2, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s0 s1 s2 ...
*** 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 2: Ejecucion de la topologia utilizando el controlador l2 learning

Aplicando el controlador, podemos notar que los 12 paquetes se reciben correctamente. Comprobado esto, procedemos a aplicar el controlador de firewall.

3.2.2. Firewall

Una vez comprobado que la topología funciona correctamente utilizando l2 learning, procedemos a modificar el controlador aplicando el firewall con una serie de reglas estipuladas, las cuales son:

1. Se deben descartar todos los mensajes cuyo puerto de destino sea 80.
2. Se deben descartar todos los mensajes que provengan del host 1, tengan el puerto de destino 5001, y esten utilizando el protocolo UDP.
3. Se deben elegir dos host cualquiera y no deben poder comunicarse.

En el archivo firewall.py se encuentran codificadas las reglas mencionadas anteriormente para el correcto filtrado de los paquetes. Para poder probar el correcto funcionamiento del controlador l2 learning con el firewall aplicado, llevaremos a cabo una serie de pasos

- Una vez ejecutada la topologia utilizando Mininet, utilizaremos iperf para realizar las correspondientes pruebas de rendimiento sobre la red. Para ello ejecutaremos el comando xterm, pasando por parámetro los host determinados.

```
*** 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:
s0 s1 s2
*** Adding links:
(h1, s0) (h2, s0) (h3, s2) (h4, s2) (s1, s0) (s2, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s0 s1 s2 ...
*** 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> xterm h1 h2
mininet>
```

Figura 3: Ejecucion de la topología y uso del comando xterm

- Luego de ejecutar el comando xterm con los host correspondientes, se abrira una consola por cada uno de ellos. A continuacioó, seleccionamos un host como servidor y ejecutamos el comando "sudo iperf -s -p <port>", y seleccionamos al otro host como cliente, ejecutando el comando "sudo iperf -c <direccionDestino>-p <puerto-Destino>". Cuando el cliente envía los paquetes al servidor, en el caso de satisfacer

alguna de las reglas mencionadas anteriormente, el firewall se encargará de filtrar los paquetes. A continuación, brindamos capturas de los casos posibles.

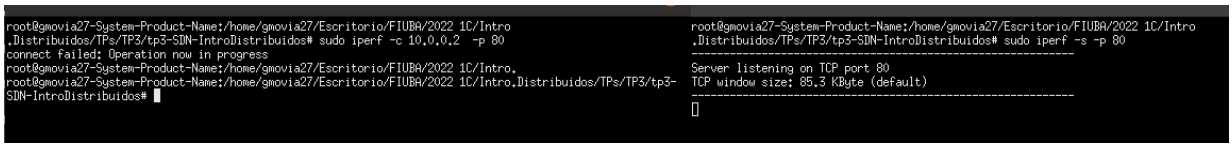


Figura 4: H1 envía un paquete a H2, y como posee el puerto 80 entonces el firewall lo filtra.

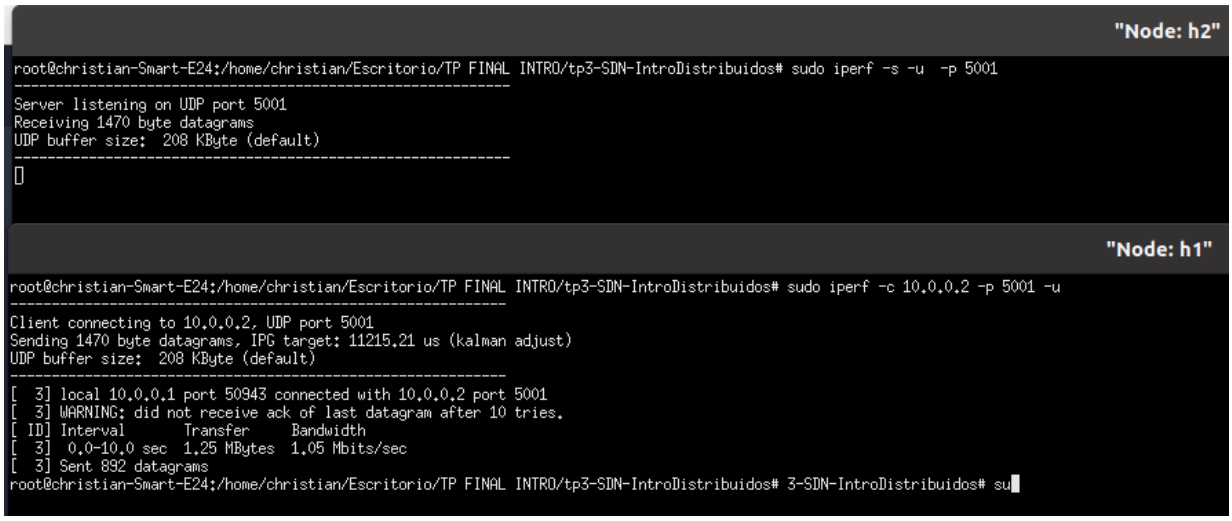


Figura 5: H1 envía un paquete a H2 utilizando el protocolo UDP, pero como H2 posee el puerto 5001, entonces el firewall lo filtra

En el "Node: h1"de la **Figura 4** podemos observar un **WARNING** que no se ha recibido un ack después de 10 intentos, esto sucedios porque la regla de Firewall ha sido aplicada en forma correcta.

4. Hipótesis y suposiciones realizadas

5. Pruebas

5.1. Servidor con puerto 80

5.1.1. TCP

Cuando establecemos una conexion TCP con un servidor que escucha en el puerto 80, lo que sucedera es que el firewall filtrara los paquetes y no llegaran al servidor, por lo que

no se podra establecer el acuerdo de tres fases, y por lo tanto, no se enviara ningun paquete de datos ya que la conexion sera fallida. Esto lo podemos observar a continuacion en la figura 6.

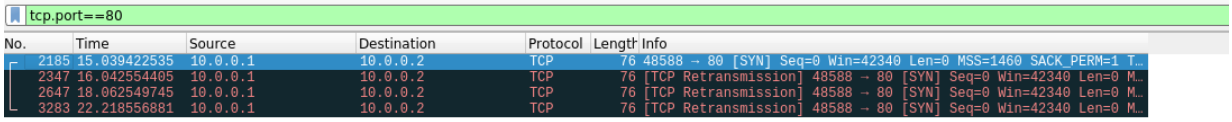


Figura 6: Se filtran los paquetes TCP con puerto 80.

5.1.2. UDP

Cuando enviamos paquetes de datos via UDP a un servidor que escucha en el puerto 80, lo que sucedera es que el firewall filtrara los paquetes y no llegaran al servidor, por lo que no se recibiran los correspondientes ACK. Esto lo podemos observar a continuacion en la figura 7.

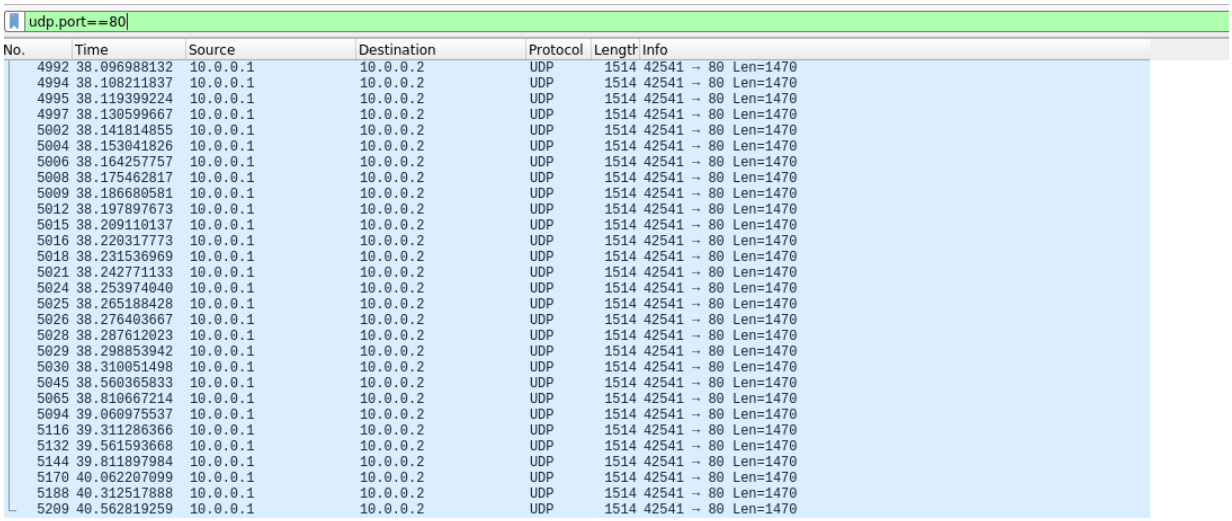
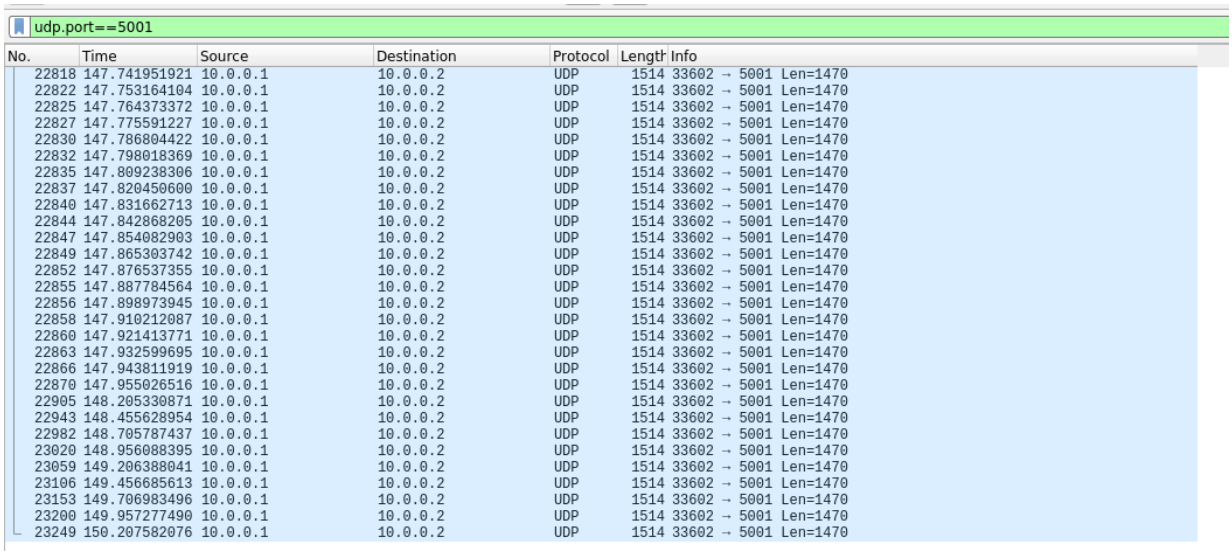


Figura 7: Se filtran los paquetes UDP con puerto 80.

5.2. Servidor con puerto 5001

Si enviamos paquetes de datos desde el host 1 via UDP a un servidor que escucha en el puerto 5001, lo que sucedera es que el firewall filtrara los paquetes y no llegaran al servidor, ya que matchean con la segunda regla establecida. Podemos notar de la figura 8 que no se reciben los ACK.



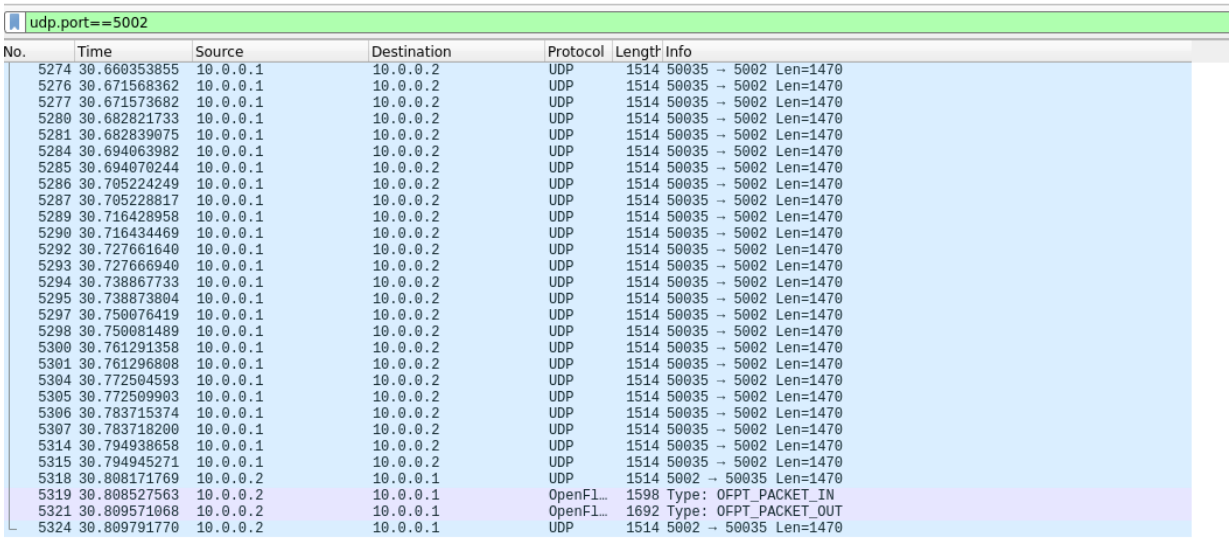
udp.port==5001

No.	Time	Source	Destination	Protocol	Length	Info
22818	147.741951921	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22822	147.753164104	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22825	147.764373372	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22827	147.775591227	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22830	147.786804422	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22832	147.798018369	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22835	147.809238306	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22837	147.820450600	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22840	147.831662713	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22844	147.842868205	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22847	147.854082903	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22849	147.865303742	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22852	147.876537355	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22855	147.887784564	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22856	147.898973945	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22858	147.910212087	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22860	147.921413771	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22863	147.932599695	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22866	147.943811919	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22870	147.955026516	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22905	148.205330871	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22943	148.455628954	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
22982	148.705787437	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23020	148.956088395	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23059	149.206388041	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23106	149.456685613	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23153	149.706983496	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23200	149.957277490	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470
23249	150.207582076	10.0.0.1	10.0.0.2	UDP	1514	33602 → 5001 Len=1470

Figura 8: Se filtran los paquetes UDP con puerto 5001.

5.3. Servidor con puerto 5002

En este caso el host 1 envia paquetes de datos via UDP a un servidor que escucha en el puerto 5002. Como los paquetes no satisfacen ninguna regla del firewall, entonces no se filtraran y llegaran correctamente al servidor. Podemos notar de la figura 9, que el servidor recibe los paquetes y envia los ACK de confirmacion nuevamente al host 1, que tiene asociado el puerto 50035.



udp.port==5002

No.	Time	Source	Destination	Protocol	Length	Info
5274	30.660353855	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5276	30.671568362	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5277	30.671573682	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5280	30.682821733	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5281	30.682839075	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5284	30.694063982	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5285	30.694070244	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5286	30.705224249	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5287	30.705228817	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5289	30.716428958	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5290	30.716434469	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5292	30.727661640	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5293	30.727666940	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5294	30.738867733	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5295	30.738873804	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5297	30.750076419	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5298	30.750081489	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5300	30.761291358	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5301	30.761296808	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5304	30.772504593	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5305	30.772509903	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5306	30.783715374	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5307	30.783718200	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5314	30.794938658	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5315	30.794945271	10.0.0.1	10.0.0.2	UDP	1514	50035 → 5002 Len=1470
5318	30.808171769	10.0.0.2	10.0.0.1	UDP	1514	5002 → 50035 Len=1470
5319	30.808527563	10.0.0.2	10.0.0.1	OpenFl…	1598	Type: OFPT_PACKET_IN
5321	30.809571068	10.0.0.2	10.0.0.1	OpenFl…	1692	Type: OFPT_PACKET_OUT
5324	30.809791770	10.0.0.2	10.0.0.1	UDP	1514	5002 → 50035 Len=1470

Figura 9: Los paquetes llegan correctamente al servidor y se reciben los ACK.

6. Dificultades encontradas

Las dificultades encontradas a lo largo del trabajo practico fueron las siguientes:

1. Dificultad a la hora de probar las reglas del firewall implementadas en el controlador.
2. Instalacion de las herramientas utilizadas.
3. Debido a que las tareas que se debian desarrollar a lo largo del trabajo estaban fuertemente relacionadas, nos encontramos con la dificultad de poder dividirlas entre los integrantes del grupo.

7. Preguntas a responder

- 1) ¿Cuál es la diferencia entre un Switch y un router? ¿Qué tienen en común?
- 2) ¿Cuál es la diferencia entre un Switch convencional y un Switch OpenFlow?
- 3) ¿Se pueden reemplazar todos los routers de la Intenet por Switches OpenFlow? Piense en el escenario interesante para elaborar su respuesta
- 1) La diferencia general entre estos dos dispositivos es que el Switch se encarga de crear una red conectando diferentes dispositivos entre ellos y el router establece la conexión entre estas redes.

  En cuanto a lo que tienen en común, ambos son dispositivos de red que permiten que una o más computadoras se puedan conectar con redes u otras computadoras.
- 2) La diferencia consiste en que un switch convencional trabaja independientemente del resto de la red, los Switch Openflow en cambio, al recibir un paquete que no tiene un flujo al cual dirigirlo, se contacta con el controlador del SDN y le pregunta qué hacer con este paquete. A partir de ahí el controlador descarga un flujo por el cual mandarlo, para poder manejar este paquete.
- 3) A pesar de ser una buena opción, existen varias razones por la cual no se reemplazan todos los routers por switches OpenFlow. La primera es una razón económica ya que está tan instalado el uso de router que sería muy costoso realizar todo el cambio.

  Otra razón es el rendimiento ya que los routers tienen un tiempo de respuesta rápida en base a la IP, mientras que OpenFlow no trabaja con IPs, sino que utiliza reglas como las utilizadas en el código de Firewall en donde solo conoce el origen y no la IP. Esto es importante ya que generaría tablas muy grandes que se deberían consultar para lograr el tráfico.

  También hay que tener en cuenta la seguridad, ya que podría existir un "man in the middle"que los switches OpenFlow no contemplan.

  Un caso interesante para la utilización de switches OpenFlow podrían ser llamadas de voz o videoconferencias.

8. Conclusión

El presente trabajo practico permitio interiorizar nuestro conocimiento en nuevas tecnologias, entre ellas destacamos: Mininet, Pox, y Wireshark.

Además nos permitió comprender con más detalle el funcionamiento de switches de OpenFlow y de cómo se comunican las redes entre sí, permitiendo configurar distintas reglas para filtrar conexiones que consideremos.

Desde el punto de vista técnico pudimos corroborar lo que hicimos a partir de distintas capturas de Wireshark y observarlo también en la sección de pruebas.