



---

# Trabajo Práctico 2: Software-Defined Networks

---

## Introducción a los sistemas distribuidos (75.43)

### Docentes:

Juan Ignacio López Lecora.

Agustín Horn.

### Integrantes:

Francisco Florit - 104289

Amaru Gabriel Durán - 97013

Valentín Flores - 107719

Federico Jaleh - 105553

Gastón Frenkel - 107718

### Fecha de entrega

18 de Junio de 2024

# Introducción

El trabajo práctico tiene como objetivo Adquirir conocimientos y práctica sobre las Software Defined Networks(SDN) y OpenFlow, como así también en el uso de herramientas de simulación de redes como mininet.

Para ello se tuvo como objetivo implementar una topología dinámica, donde a través de OpenFlow se construyó un firewall a nivel capa de enlace.

## Objetivo

Se propone desarrollar una topología parametrizable sobre la cual probaremos diferentes funcionalidades que nos brinda la tecnología OpenFlow.

Se tendrá una cantidad de switches variable, formando una cadena, en cuyos extremos se tienen dos hosts. La topología debe recibir por parámetro la cantidad de switches.

Una vez lograda la topología, se debe verificar el correcto funcionamiento de la red, a través del comando pingall. Se debe correr el comando pingall en el entorno de mininet, y poder registrar el envío y recepción de los mensajes, tanto en Wireshark como en los registros del controlador remoto.

Una vez verificado el correcto funcionamiento de la red, vamos a modificar el controlador, para que funcione como un Firewall, con una serie de reglas definidas a continuación:

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

## Documentación (Readme)

### Requisitos de instalación

- Python version 3.7|3.8|3.9
- Tener instalado mininet.
- Tener instalado Openvswitch.

### Levantar servidor pox con firewall sobre OpenFlow

1. Dar permisos de ejecución al script `run_server.sh`:

```
chmod -x run_server.sh
```

2. Ejecutar con:

```
./run_firewall.sh
```

## Levantar topología con switchs variable

1. Dar permisos de ejecución al script `run_topo.sh`:

```
chmod -x run_topo.sh
```

2. Ejecutar con:

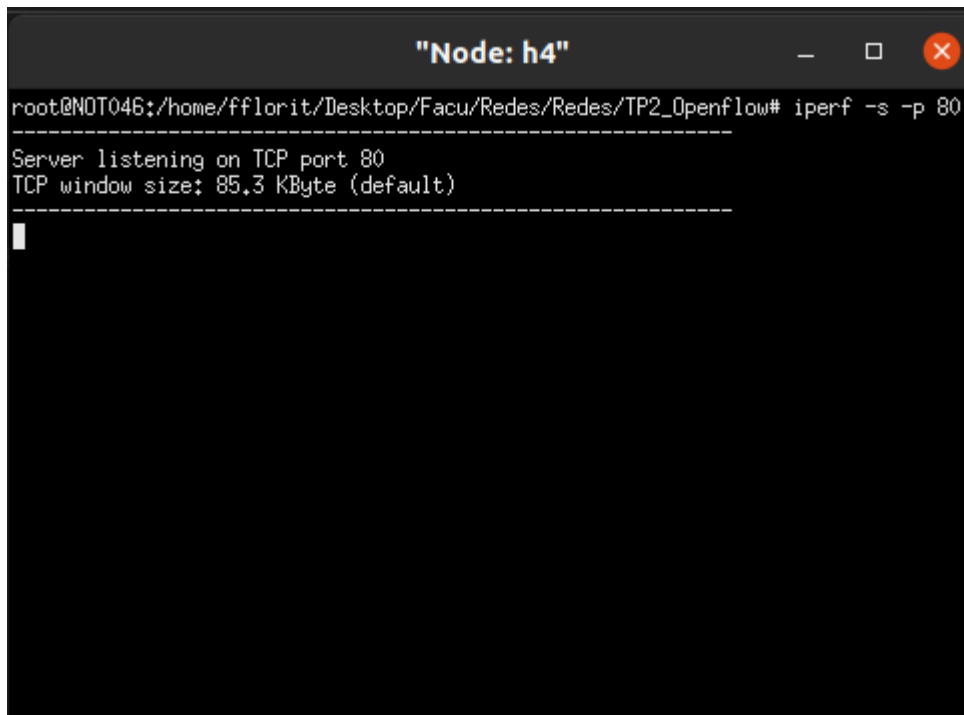
```
./run_topo.sh <cantidad de switches>
```

## Pruebas

A continuación se detallan algunas de las pruebas realizadas para verificar el correcto funcionamiento del firewall:

1. descartar todos los mensajes cuyo puerto destino sea 80.

Se puede observar que al iniciar el servidor con puerto 80 utilizando el protocolo tcp en h4:



```
"Node: h4"
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow# iperf -s -p 80
-----
Server listening on TCP port 80
TCP window size: 85,3 KByte (default)
-----
█
```

Al momento de generar una conexión con un cliente en h2 se puede ver que se rechazan las conexiones provenientes de h2.

```
"Node: h2"
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow# iperf -c 10.0.0.4 -p 80
connect failed: Operation now in progress
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow#
```

Al iniciar el servidor con puerto distinto de 80, en este caso con un puerto 12345:

```
"Node: h4"
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow# iperf -s -p 12345
-----
Server listening on TCP port 12345
TCP window size: 85,3 KByte (default)
-----
```

Al momento de generar una conexión con un cliente en h2 se puede ver que se envían los mensajes sin problemas.

```
"Node: h2"
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow# iperf -c 10.0.0.4 -p 12345
-----
Client connecting to 10.0.0.4, TCP port 12345
TCP window size: 1.72 MByte (default)
-----
[ 23] local 10.0.0.2 port 55606 connected with 10.0.0.4 port 12345
[ ID] Interval      Transfer    Bandwidth
[ 23] 0.0-10.0 sec  37.5 GBytes 32.2 Gbits/sec
root@NOT046:/home/fflorit/Desktop/Facu/Redes/Redes/TP2_Openflow#
```

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

Se puede observar que al iniciar el servidor con puerto 5001 utilizando el protocolo udp con el flag **-u** en h4:

```
"Node: h4"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -i1 -s -u -p 5001
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
█
```

Al momento de generar una conexión con un cliente en h1 se puede ver que se rechazan las conexiones provenientes de h1.

```
"Node: h1"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -c 10.0.0.4 -p 5001 -u
-----
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)
-----
[ 23] local 10.0.0.1 port 51334 connected with 10.0.0.4 port 5001
[ 23] WARNING: did not receive ack of last datagram after 10 tries.
[ ID] Interval      Transfer    Bandwidth
[ 23] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 23] Sent 892 datagrams
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow#
```

Mientras que si por ejemplo generamos un cliente desde h2 esto no sucede y se envían los mensajes sin problemas:

```
"Node: h2"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -c 10.0.0.4
-p 5001 -u
-----
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)
-----
[ 23] local 10.0.0.2 port 45359 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 23] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 23] Sent 892 datagrams
[ 23] Server Report:
[ 23] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.005 ms  0/ 892 (0%)
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow#
```

servidor h4:

```
"Node: h4"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -i1 -s -u -p 5001
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 23] local 10.0.0.4 port 5001 connected with 10.0.0.2 port 45359
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams
[ 23] 0.0- 1.0 sec    129 KBytes    1.06 Mbits/sec  0.002 ms   0/ 90 (0%)
[ 23] 1.0- 2.0 sec    128 KBytes    1.05 Mbits/sec  0.004 ms   0/ 89 (0%)
[ 23] 2.0- 3.0 sec    128 KBytes    1.05 Mbits/sec  0.005 ms   0/ 89 (0%)
[ 23] 3.0- 4.0 sec    129 KBytes    1.06 Mbits/sec  0.004 ms   0/ 90 (0%)
[ 23] 4.0- 5.0 sec    128 KBytes    1.05 Mbits/sec  0.006 ms   0/ 89 (0%)
[ 23] 5.0- 6.0 sec    128 KBytes    1.05 Mbits/sec  0.004 ms   0/ 89 (0%)
[ 23] 6.0- 7.0 sec    128 KBytes    1.05 Mbits/sec  0.004 ms   0/ 89 (0%)
[ 23] 7.0- 8.0 sec    128 KBytes    1.05 Mbits/sec  0.006 ms   0/ 89 (0%)
[ 23] 8.0- 9.0 sec    128 KBytes    1.05 Mbits/sec  0.003 ms   0/ 89 (0%)
[ 23] 0.0-10.0 sec    1.25 MBytes    1.05 Mbits/sec  0.005 ms   0/ 892 (0%)
```

### 3. Dos hosts no se pueden comunicar.

En nuestro caso elegimos los hosts 1 y 3, los cuales no se pueden comunicar:

#### Caso UDP

Levantamos un server en h1 y se verifica que desde h3 no se le puede enviar mensajes:

```
"Node: h3"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -c 10.0.0.1 -p 3500 -u
-----
Client connecting to 10.0.0.1, UDP port 3500
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 23] local 10.0.0.3 port 42255 connected with 10.0.0.1 port 3500
[ 23] WARNING: did not receive ack of last datagram after 10 tries.
[ ID] Interval      Transfer      Bandwidth
[ 23] 0.0-10.0 sec    1.25 MBytes    1.05 Mbits/sec
[ 23] Sent 892 datagrams
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow#

"Node: h1"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_Openflow# iperf -s -p 3500 -u
-----
Server listening on UDP port 3500
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
```

Levantamos server en h3 y se verifica que desde h1 no se le puede enviar mensajes:

```
"Node: h3"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow# iperf -s -p 4000 -u
Server listening on UDP port 4000
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

"Node: h1"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow# iperf -c 10.0.0.3 -p 4000 -u
Client connecting to 10.0.0.3, UDP port 4000
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 208 KByte (default)

[ 23] local 10.0.0.1 port 53220 connected with 10.0.0.3 port 4000
[ 23] WARNING: did not receive ack of last datagram after 10 tries.
[ ID] Interval      Transfer      Bandwidth
[ 23] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 23] Sent 892 datagrams
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow#
```

## Caso TCP

Levantamos un server TCP en h1 y se verifica que desde h3 no se le puede enviar mensajes:

```
"Node: h3"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow# iperf -c 10.0.0.1 -p 5000
connect failed: Operation now in progress
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow#

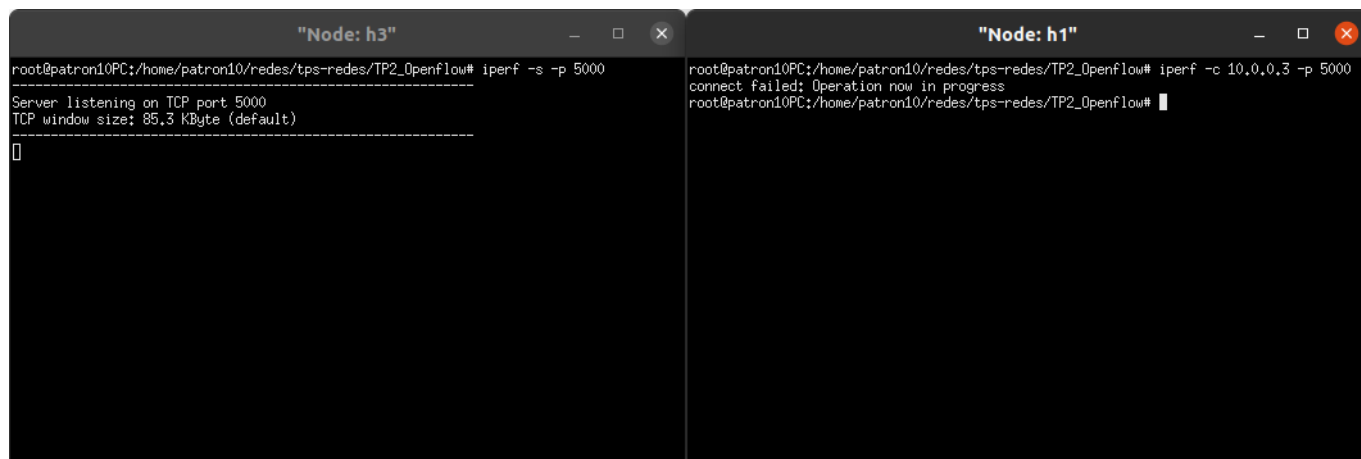
"Node: h1"
root@patron10PC:/home/patron10/redes/tps-redes/TP2_0penflow# iperf -s -p 5000
Server listening on TCP port 5000
TCP window size: 85.3 KByte (default)
```

Con wireshark se puede ver que en el switch 1, que es donde tenemos instalado el firewall, figura que se pierden paquetes por retransmisión de SYN:

Capturando desde s1-eth2						
Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda						
Aplique un filtro de visualización ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Sequence Info
1	0.000000000	10.0.0.3	10.0.0.1	TCP	74	44740 → 5000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460
2	1.021627337	10.0.0.3	10.0.0.1	TCP	74	[TCP Retransmission] 44740 → 5000 [SYN] Seq=0 Win=
3	3.037155361	10.0.0.3	10.0.0.1	TCP	74	[TCP Retransmission] 44740 → 5000 [SYN] Seq=0 Win=
4	7.101311079	10.0.0.3	10.0.0.1	TCP	74	[TCP Retransmission] 44740 → 5000 [SYN] Seq=0 Win=
5	51.645327821	fe80::9862:47ff:fe5...	ff02::2	ICMPv6	70	Router Solicitation from 9a:62:47:5d:fd:b5
6	51.649218381	fe80::200:ff:fe00:3	ff02::2	ICMPv6	70	Router Solicitation from 00:00:00:00:00:03

Ahora, Levantamos server en h3 y se verifica que desde h1 no se le puede enviar mensajes:





Mismo análisis visto desde wireshark, pero desde el link de salida a h1:

Capturando desde s1-eth1						
Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda						
Aplique un filtro de visualización ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Sequen Info
1	0.000000000	10.0.0.1	10.0.0.3	TCP	74	40576 → 5000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460
2	1.007091695	10.0.0.1	10.0.0.3	TCP	74	[TCP Retransmission] 40576 → 5000 [SYN] Seq=0 Win=
3	3.018836416	10.0.0.1	10.0.0.3	TCP	74	[TCP Retransmission] 40576 → 5000 [SYN] Seq=0 Win=
4	7.146877522	10.0.0.1	10.0.0.3	TCP	74	[TCP Retransmission] 40576 → 5000 [SYN] Seq=0 Win=

## Preguntas a responder

### ***¿Cuál es la diferencia entre un Switch y un router? ¿Qué tienen en común?***

Tanto los routers como los switches se dedican a redireccionar datos en una red para que lleguen de un origen a un destino. La principal diferencia es que los routers conectan redes entre sí utilizando las direcciones IP del destino y los switches conectan dispositivos dentro de la misma red local, utilizando las direcciones MAC.

### ***¿Cuál es la diferencia entre un Switch convencional y un Switch OpenFlow?***

La diferencia entre un switch openflow y uno convencional radica en el control de los mismos.

Los switch openflow tienen un control centralizado donde la lógica de red se separa del hardware y se maneja desde el "control plane" del SDN, es desde este lugar del que se toma la decisiones de filtrado de paquetes, eleccion de flujo del paquete, etc. Sin embargo, los switch tradicionales tienen un control distribuido y cada switch, en el propio hardware, manejan su propia lógica de control basados en sus tablas de direcciones Mac y políticas de configuración para decidir si hay que filtrar algún paquete.

### ***¿Se pueden reemplazar todos los routers de la Intenet por Switches OpenFlow? Piense en el escenario interASes para elaborar su respuesta***

Teniendo en cuenta los escenarios SA, los routers están preparados para gestionar tráfico a una gran escala dado que utilizan hardware especializado (ASICs) para procesar rápidamente los paquetes, sin embargo en los casos de los switches openFlow requieren una conexión constante con el plano de control para tomar decisiones de enrutamiento, acción que podría agregar latencia. Además, los routers utilizan protocolos más complejos que los utilizados por los switches, como por ejemplo el protocolo de enrutamiento BGP (Border Gateway Protocol), utilizado precisamente a la hora de mantener conexiones entre distintos sistemas autónomos (AS).

## Dificultades encontradas

- Instalar dependencias.
- Setear el firewall con OpenFlow.
- Fix en código de Pox para python3.

## Conclusiones

El trabajo práctico nos permitió entender más en profundidad como funcionan las "Software Defined Networks" y darnos la experiencia de poder separar el plano de datos, donde se encuentran los switches generados por nuestra topología lineal de N switches variables, del plano de control, de forma que pudimos programar en el mismo la lógica que nos permite decidir qué paquetes filtrar a través del switch que tiene configurado un firewall, utilizando una serie de reglas definidas por nosotros mismos.