

# Trabajo Práctico 2: Software-Defined Networks

Facultad de Ingeniería de la Universidad de  
Buenos Aires

Redes

Cátedra Hamelin-Lopez Pecora



Demarchi, Ignacio  
Padrón: 107835  
email: idemarchi@fi.uba.ar

Lijs, Theo  
Padrón: 109472  
email: tlijs@fi.uba.ar

Schneider, Valentin  
Padrón: 107964  
email: vschneider@fi.uba.ar

Orsi, Tomas Fabrizio  
Padrón: 109735  
email: torsi@fi.uba.ar

## Contents

## **1 Introducción**

En este trabajo practico se implemento la elaboracion de un SDN que, mediante OpenFlow (utilizando POX) implementa un Firewall sobre una red creada en Mininet. Para poder ver el programa en accion y acercar la simulacion a un caso de uso real, dentro de los hosts de la red de Mininet se utiliza iperf para establecer facilmente una conexion entre clientes y servidores y ver el funcionamiento del Firewall en accion. Para poder realmente ver esto, se utiliza Wireshark donde se observan los paquetes siendo enviados.

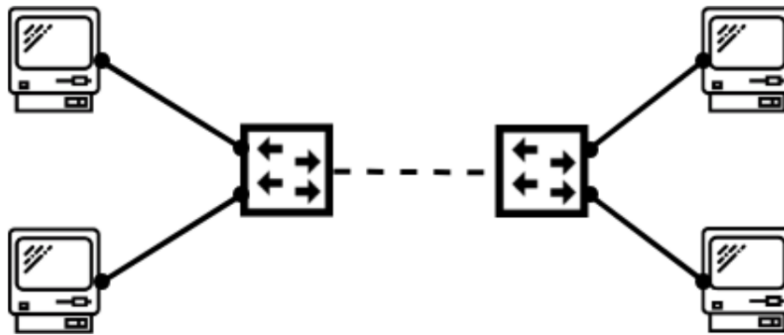
## **2 Hipótesis y suposiciones realizadas**

## 3 Herramientas utilizadas

A continuacion se detalla el uso de cada herramienta ya mencionada para elaborar el trabajo practico.

### 3.1 Mininet

Para utilizar Mininet, la topologia se define en mytopo.py. La misma recibe como parametro la cantidad de switches a utilizar.



Al correr el comando para levantar mininet, se establece la ip del controlador que se va a utilizar. Esto es para que luego cuando corramos el controlador el mismo pueda modificar los switches de la topologia y maneje el control plane de la red de mininet.

### 3.2 POX WIPP

explicar como es que levanta y aplica las reglas, el formato de las reglas y como conecta a lo que este corriendo en mininet

Para utilizar OpenFlow, se usa POX.

Explicar lo de l2 learning en POX

Mostrar que los logs y wireshark muestran masomenos lo mismo

Es importante destacar que estos switches no aprenden, por lo cual sin interacción con el controlador, nunca enviarán un paquete. Dado que OpenFlow se enfoca en poder controlar los flujos, los switches que manejan este protocolo pueden ejecutar múltiples acciones según si los paquetes coinciden con una entrada en la tabla de flujos.

### 3.3 Wireshark & iperf

Para comprobar el correcto funcionamiento de la red y del Firewall, se utiliza iperf para simular clientes y servidores, y Wireshark para ver los paquetes enviados.

Al tener abierta la red de mininet, Wireshark la detecta como opcion para poder escuchar.

#### IMAGEN REDES QUE VE WIRESHARK

A continuacion se muestra el correcto funcionamiento del controlador, usando pingall dentro de mininet y escuchando con Wireshark.

```
o valen1611@ubuntu1611:~/code/redes$ sudo make mininet
mn --custom src/mytopo.py --topo mytopo,7 --controller remote,i
p=127.0.0.1,port=6633
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
mininet1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, mininet1) (h2, mininet1) (h3, s7) (h4, s7) (mininet1, s2) (s
2, s3) (s3, s4) (s4, s5) (s5, s6) (s6, s7)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 7 switches
mininet1 s2 s3 s4 s5 s6 s7 ...
*** 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>

o valen1611@ubuntu1611:~/code/redes$ make run
sed -i 's/IPDELSWITCHCONELFIREWALL = [0-9]/IPDELSWITCHCONELFIREWALL = 1/' src/fir
ewall.py
pox/pox.py firewall forwarding.l2_learning log.level --DEBUG samples.pretty_log
POX 0.6.0 (fangtooth) / Copyright 2011-2018 James McCauley, et al.
[core] POX 0.6.0 (fangtooth) going up...
[core] Running on CPython (2.7.18/Oct 15 2023 16:43:11)
[core] Platform is Linux-6.5.0-1024-oem-x86_64-with-Ubuntu-22.
04-jammy
[core] POX 0.6.0 (fangtooth) is up.
[openflow.of_01] Listening on 0.0.0.0:6633
[openflow.of_01] Connection [00-00-00-00-00-03 5] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-03 5]
[openflow.of_01] Connection [00-00-00-00-00-06 4] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-06 4]
[openflow.of_01] Connection [00-00-00-00-00-05 7] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-05 7]
[openflow.of_01] Connection [00-00-00-00-00-04 3] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-04 3]
[openflow.of_01] Connection [00-00-00-00-00-07 2] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-07 2]
[openflow.of_01] Connection [00-00-00-00-00-01 1] connected
[firewall] Firewall rules installed on 00-00-00-00-00-01
[forwarding.l2_learning] Connection [00-00-00-00-00-01 1]
[openflow.of_01] Connection [00-00-00-00-00-02 6] connected
[forwarding.l2_learning] Connection [00-00-00-00-00-02 6]
[forwarding.l2_learning] installing flow for 5a:e2:f1:fa:d2:51.2 -> 16:95:7d:78:
e4:be.1
[forwarding.l2_learning] installing flow for 16:95:7d:78:e4:be.1 -> 5a:e2:f1:fa:
d2:51.2
[forwarding.l2_learning] installing flow for 5a:e2:f1:fa:d2:51.2 -> 16:95:7d:78:
e4:be.1
[forwarding.l2_learning] installing flow for 2e:91:98:b2:56:33.2 -> 16:95:7d:78:
e4:be.1
```

#### IMAGEN TRAZA WIRESHARK CON EL PINGALL DE MININET

Para poder comprobar que no solo funciona con ICMP utilizamos iperf. Con iperf simulamos clientes y servidores. En el ejemplo a continuacion tenemos al host h1 actuando como cliente y al host h3 actuando como servidor, comunicandose por TCP.

```
"Node: h1"
root@ubuntu1611:/home/valen1611/code/redis# sudo iperf -c 10.0.0.3 -p 80
Client connecting to 10.0.0.3, TCP port 80
TCP window size: 85.3 KByte (default)
[ 1] local 10.0.0.1 port 50136 connected with 10.0.0.3 port 80
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0193 sec 17.8 GBytes 15.3 Gbits/sec
root@ubuntu1611:/home/valen1611/code/redis#

"Node: h3"
root@ubuntu1611:/home/valen1611/code/redis# sudo iperf -s -p 80
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
[ 1] local 10.0.0.3 port 80 connected with 10.0.0.1 port 50136
[ ID] Interval      Transfer    Bandwidth
[ 1] 0.0000-10.0072 sec 17.8 GBytes 15.3 Gbits/sec

mininet> xterm h1 h3
mininet> 
```

IMAGEN TRAZA WIRESHARK CON EL IPERF BASICO

## 4 Resultados de simulaciones WIP FALTAN IM- AGENES

### 4.1 Puerto Destino 80

Simulacion para descartar todos los mensajes cuyo puerto destino sea 80.

#### 4.1.1 Reglas

```
{
  "policies":[
    {
      "dst_port": "80"
    }
  ]
}
```

#### 4.1.2 Wireshark

#### 4.1.3 Logs del controlador

### 4.2 Host 1, Puerto 5001 y UDP

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

#### 4.2.1 Reglas

```
{
  "policies":[
    {
      "src_ip": "10.0.0.1",
      "dst_port": 5001,
      "protocol": "UDP"
    }
  ]
}
```

#### 4.2.2 Wireshark

#### **4.2.3 Logs del controlador**

### **4.3 Dos hosts no se comunican entre si**

Simulacion donde se eligen dos hosts cualquiera, y los mismos no pueden comunicarse de ninguna forma.

#### **4.3.1 Reglas**

```
{
  "policies":[
    {
      "banned_tuples": ["10.0.0.1","10.0.0.3"]
    }
  ]
}
```

#### **4.3.2 Wireshark**

#### **4.3.3 Logs del controlador**



## **5 Preguntas a responder**

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

La principal diferencia es que un switch opera en la capa 2 (enlace) y un router en la capa 3 (red). Los switches redireccionan utilizando la dirección MAC de los dispositivos, mientras que los routers utilizan la IP.

Lo que tienen en común es que ambos funcionan para redireccionar paquetes y permitir que hosts en distintas partes del mundo puedan comunicarse entre sí.

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

La diferencia más importante entre un Switch convencional y uno OpenFlow es que el OpenFlow puede ser gestionado mediante software con un controlador centralizado. Lo que permite automatizar y agilizar el proceso. Los switches convencionales no tienen el plano de control desacoplados por lo que configurarlos requiere más trabajo.

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

No es algo factible, ya que

## **6 Dificultades encontradas**

## **7 Conclusión**