Anexo 1

Guión de prácticas de laboratorio para impartición docente

Sergio Rodríguez Santamaría

Práctica *

Software-Defined Networking

OpenFlow y Mininet.

Las tecnologías de telecomunicación, sobra decirlo, están evolucionando a un ritmo vertiginoso. Como usuarios finales, percibimos dichos cambios en mayor medida en aquellos dispositivos que empleamos de manera más habitual, como pueden ser los teléfonos móviles, las tabletas o los ordenadores. Sin embargo, no es menos cierto que también se está produciendo una notable mejoría de las arquitecturas de red que se utilizan en la parte dorsal. No sólo se trata de un mero avance en su capacidad, habitualmente derivado del uso de tecnología óptica, sino que se está también modificando la filosofía en los procedimientos y algoritmos de encaminamiento.

En este marco, una de las iniciativas que mayor interés ha suscitado recientemente es OpenFlow [1]. Se trata de una propuesta que facilita la evolución de las redes, a través de un controlador remoto que es capaz de modificar el comportamiento de los dispositivos de red a través de un conjunto de instrucciones bien definido. Incorpora diferentes tipos de nodos: conmutadores, encaminadores, etc. de varios fabricantes.

[1] http://www.openflow.org/

Objetivo de la práctica:

El objetivo de esta práctica es el de familiarizarse en primer lugar con todo lo que conlleva OpenFlow, analizar sus posibilidades, etc. Posteriormente, se planteará el uso de la herramienta Mininet, que permite utilizar modelos de redes en un ordenador (de manera virtual) para probar algunas de las funcionalidades principales de OpenFlow.

Antes de comenzar, el profesor llevará a cabo una presentación del protocolo OpenFlow y de la herramienta Mininet, comentando los aspectos principales de ambos. Además, se explicará el entorno de trabajo para el desarrollo de la práctica.

La práctica está dividida en tres partes y en cada una se tratarán los siguientes temas:

Parte 1: Estudio y análisis de una topología de red básica.

Parte 2: Implementación de una red en código Python.

Parte 3: Estudio de las implementaciones GUI facilitadas por Mininet.

Para realizar esta práctica deberéis dividiros en parejas y la evaluación de la práctica se hará in-situ y consistirá en la comprobación por parte del profesor de que se han realizado con éxito todos los ejercicios propuestos en cada apartado.

Parte 1: Estudio y análisis de una topología de red básica.

INTRODUCCIÓN

Tras introducir brevemente el entorno sobre el que se trabaja, para esta primera parte de la practica será necesario utilizar dos herramientas:

1. WireShark

En primer lugar, para poder realizar un estudio del tráfico generado por una red, es necesario comentar el modo de uso de la herramienta WireShark dentro de Mininet. Para su ejecución simplemente será necesario introducir la instrucción **"sudo wireshark &"**, de esta manera se abrirá el programa y se ejecutará en segundo plano. Dentro de WireShark será necesario aplicar el filtro correcto, para ello, se utilizará el filtro 'ofp' (OpenFlow Protocol). Esto permitirá filtrar los paquetes OpenFlow generados en la red. El último paso para comenzar a capturar paquetes será seleccionar la interfaz 'loopback' (lo). Hecho esto, ya será posible comenzar a trabajar con Mininet visualizando el tráfico generado en una red de manera correcta.



Capture					
Live list of the capture interfaces (counts incoming packets)					
Start capture on interface:					
 eth0 Pseudo-device that captures on all interfaces USB bus number 1 					
Io					

2. CLI Mininet

El CLI de Mininet es una interfaz por línea de comandos que ofrece un mecanismo a través del cual resulta mas sencillo comunicarse con los distintos nodos de una red.



PRÁCTICA

Una vez ubicados, en esta primera parte de la práctica el objetivo es familiarizarse con las principales funciones de Mininet.

Para ello siguiendo el tutorial facilitado en la página principal de Mininet [2] se pide llevar a cabo las siguientes tareas.

[2] http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet

Ejercicio 1.1:

Obtención de información de los distintos nodos de la red

Ejercicio 1.2:

Realización de los siguientes test sobre la topología:

- 1. ping
- 2. pingAll
- 3. pingPair
- 4. iperf

Ejercicio 1.3:

Análisis de los resultados con la herramienta WireShark

Parte 2: Implementación de una red en código Python

INTRODUCCIÓN

Cualquier topología de red puede ser fácilmente diseñada e implementada en Mininet, mediante el uso de una API en Python.

Una vez completada la primera parte de la práctica, el principal objetivo en esta segunda parte es **aprender a implementar una red, compilarla y ejecutarla correctamente**.

Para ello se puede partir del ejemplo que se adjunta:

1. Explicación de un código ejemplo.

En primer lugar es necesario comentar la manera en la que se programa una red básica para su posterior uso en Mininet. A continuación se puede observar el código Python de una red ejemplo, la cual consta de dos switch y dos hosts.

```
from mininet.topo import Topo, Node
class MyTopo( Topo ):
    "Simple topology example."
    def __init__( self, enable_all = True ):
        "Create custom topo."
        # Add default members to class.
        super( MyTopo, self ).__init__()
        # Set Node IDs for hosts and switches
        leftHost = 1
        leftSwitch = 2
        rightSwitch = 3
        rightHost = 4
        # Add nodes
        self.add_node( leftSwitch, Node( is_switch=True ) )
        self.add_node( rightSwitch, Node( is_switch=True ) )
self.add_node( leftHost, Node( is_switch=False ) )
        self.add_node( rightHost, Node( is_switch=False ) )
        # Add edges
        self.add_edge( leftHost, leftSwitch )
        self.add_edge( leftSwitch, rightSwitch )
        self.add_edge( rightSwitch, rightHost )
        # Consider all switches and hosts 'on'
        self.enable_all()
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Como se observa en las primeras líneas, se importan los paquetes necesarios, se crea la clase y se gestiona el usuario de la red.

```
from mininet.topo import Topo, Node
class MyTopo( Topo ):
    "Simple topology example."
    def __init__( self, enable_all = True ):
        "Create custom topo."
        # Add default members to class.
        super( MyTopo, self ).__init__()
```

Los siguientes fragmentos de código se encargan de: asignar un ID a cada nodo de la red, gestionar cada uno de ellos y establecer los enlaces correspondientes.

```
# Set Node IDs for hosts and switches
leftHost = 1
leftSwitch = 2
rightSwitch = 3
rightHost = 4
# Add nodes
self.add_node( leftSwitch, Node( is_switch=True ) )
self.add_node( rightSwitch, Node( is_switch=True ) )
self.add_node( leftHost, Node( is_switch=False ) )
self.add_node( rightHost, Node( is_switch=False ) )
# Add edges
self.add_edge( leftHost, leftSwitch )
self.add_edge( leftSwitch, rightSwitch )
self.add_edge( rightSwitch, rightHost )
```

Por último se habilitan todos los nodos generados y se nombra la topología creada para su posterior llamada desde Mininet.

```
# Consider all switches and hosts 'on'
self.enable_all()
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

2. Compilación y ejecución del fichero .py generado.

Una vez tengamos el fichero .py el siguiente paso es transferir ese fichero a la máquina virtual para posteriormente compilarle y ejecutarle.

Para pasar el archivo a la maquina virtual es necesario emplear la instrucción 'SCP'. SCP es un comando SSH que permite transferir archivos o carpetas entre computadores. La sintaxis es la siguiente:

\$ scp archivo usuario@servidor.com:ruta

El proceso para transferir el archivo, es el que se muestra a continuación:

En primer lugar debemos conocer la ruta donde se tendrá que ubicar el fichero.

```
openflow@openflowtutorial:~$ ls
mininet nox oflops oftest openflow openvswitch
openflow@openflowtutorial:~$ cd mininet
openflow@openflowtutorial:~/mininet$ ls
bin custom doxygen.cfg INSTALL Makefile mininet.egg-info
mnexec.c setup.py
build dist examples LICENSE mininet mnexec
README util
openflow@openflowtutorial:~/mininet$ cd custom
openflow@openflowtutorial:~/mininet/custom$ ls
README topo-2sw-2host.py
openflow@openflowtutorial:~/mininet/custom$
```

Una vez localizada, a través de la instrucción SCP comentada anteriormente, se copiará el fichero en la máquina virtual:

Equipo: Desktop user\$ scp FICHERO.py openflow@of:~/mininet/custom/ openflow@of's password:

redLineaDocente.py 100% 1389 1.4KB/s 00:00

Una vez completado ya se tiene en la ruta correcta la topología implementada.

openflow@openflowtutorial:~/mininet/custom\$ ls README **FICHERO.py** topo-2sw-2host.py

Mediante este procedimiento todas las redes que se quieran analizar pueden ser transferidas a la máquina virtual para posteriormente ejecutar pruebas sobre ellas.

Para comprobar su correcto funcionamiento, es necesario introducir la siguiente instrucción:

openflow@openflowtutorial:~/mininet/custom\$ sudo mn --custom FICHERO.py --topo NombreDado

> custom in sys.argv *** Adding controller *** Creating network *** Adding hosts: h1 h2 *** Adding switches: s3 s4 *** Adding links: (h1, s3) (s3, s4) (s4, h2) *** Configuring hosts

```
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet>
```

A continuación ya se puede comenzar a trabajar sobre la red.

PRÁCTICA

Una vez explicados los conceptos necesarios para el desarrollo de esta parte de la practica, los ejercicios propuestos son los siguientes:

Ejercicio 2.1:

Desarrollar el código en Python que implemente el siguiente esquema de red.



Figura 1, Topología a implementar.

Ejercicio 2.2:

Compilar el código y comprobar que todos los nodos son cargados en Mininet correctamente.

Ejercicio 2.3:

Realizar las pruebas de la parte 1 y comentar los resultados obtenidos:

Parte 3: Estudio de las implementaciones facilitadas por Mininet.

INTRODUCCIÓN

Existen un conjunto de herramientas, desarrolladas en Python, que tienen como objetivo servir de ayuda a la hora de comenzar a trabajar con Mininet. En esta tercera parte el objetivo será el de familiarizarse con ellas y entender su funcionamiento.

Algunas de estas implementaciones generan distintas topologías de red cada una con unas determinadas características, mientras que otras, como muestra la figura 2, generan interfaces gráficas para el usuario (GUIs) para su interacción con Mininet.

00	X N	lininet		\varTheta 🔿 🔿 📉 MiniEdit
Hosts Switches	Controllers Grap	h Ping Iperf	Interrupt Clear	MiniEdit Edit Run
ht	h2	h3	h4	
15.6 HBytes 131 Hbits/sec	14.5 HBytes 122 Hbits/sec	13.7 HBytes 115 Hbits/sec	11.7 HBytes 98.1 Hbits/sec	
h5	h6	h7	h8	
13.7 MBytes 115 Mbits/sec	14.4 MBytes 121 Mbits/sec	15.0 HBytes 126 Hbits/sec	12.2 MBytes 103 Mbits/sec	
h9	h 10	h11	h12	
14.8 HBytes 124 Hbits/sec	13.9 HBytes 116 Hbits/sec	15.3 HBytes 129 Hbits/sec	10.8 HBytes 90.4 Hbits/sec	
h13	h14	h15	h16	Stop
14.5 HBytes 121 Noits/sec	16.8 HBytes 141 Hbits/sec	14.5 HBytes 122 Hbits/sec	12.2 HBytes 103 Hbits/sec	

Figura 2, multitest.py y miniedit.py

PRÁCTICA

Los ejercicios para esta parte son los siguientes:

Ejercicio 3.1:

Ejecutar las distintas herramientas [2] y analizar cada una de ellas por separado.

[2] https://github.com/mininet/mininet/tree/master/examples

Ejercicio 3.2:

Ejecutar el fichero miniedit.py e intentar realizar la red del apartado 2.

Ejercicio 3.3:

Analizar el fichero multitest.py y comentar las distintas opciones que ofrece.