Laboratorio 3

Integrantes: Sofía Riquelme y Javier Pérez
Grupo: 11

Desarrollo de la tarea

Para el desarrollo de la tarea se instaló una máquina virtual con mininet 2.3.0 en VirtualBox. Posteriormente se hizo una carpeta compartida para poder tener archivos en python y así tener scripts con las topologías y los controladores de cada red

Red 1: Anillo Interconectado

Creación de la red

Para la creación de esta red se utilizó un script de python conteniendo lo siguiente:

```
from mininet.topo import Topo
class MyTopo(Topo):
    def __init__(self):
        Topo.__init__(self)

        sw = []
        for i in range(4):
            sw.append(self.addSwitch(f's{i+1}', dpid = f'{i+1}'))

        hsts = []
        for i in range(8):
            hsts.append(self.addHost(f'h{i+1}',mac = f'10:00:00:00:00:0{i+1}'))

        for i in range(4):
            self.addLink(sw[i], sw[i-1], 3, 4)
        self.addLink(sw[1], sw[3], 5, 6)#conexion extra

        for k in range(4):
            for i in range(2):
               self.addLink(hsts[i+(k*2)], sw[k], 100, i+1)

topos = {'MyTopo': (lambda: MyTopo())}
```

Lo que hace este script es añadir todos los hosts y los switches de forma que cada switch esté conectado a dos hosts y cada switch esté conectado entre ellos. También a cada host se le añade una dirección MAC.

Controladores

Posteriormente, se hizo un controlador para la primera topología, de forma de que todos los hosts se pudieran comunicar entre ellos. Se usó como código base de este controlador (y todos los siguientes) con el código de ejemplo de `l2_learning.py` dentro de la carpeta `pox` que viene al instalar mininet

El controlador base se modificó para hacer conexiones en ambos sentidos. Luego, se creó un segundo controlador para que la red solo funcione en sentido antihorario (ya que somos el grupo 11).

Testeo de conexiones

Luego de crear el controlador para la configuración con Opengflow, se testeó haciendo un ping hacia todos los hosts:

```
sofiwi@sofiwi-VM:~/mininet$ sudo mn --custom topologia1.py --topo MyTopo --mac --controller remote --switch ovsk
*** 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 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, s4) (s2, s1) (s2, s4) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```

Figure 1: Ping para verificar comunicación entre hosts

¿Qué sucede al eliminar una conexión?

Si eliminamos la conexión entre dos de los switches, los hosts ya no se pueden comunicar de manera correcta entre ellos. Al morir esa conexión, se puede ver que hay 4 hosts que ya no se pueden comunicar entre ellos, pero el controlador sigue manteniendo la comunicación con las conexiones que siguen funcionando:

```
mininet> link s1 s2 down
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X X X h7 h8
h2 -> h1 X X X X h7 h8
h3 -> X X h4 h5 h6 X X
h4 -> X X h3 h5 h6 X X
h5 -> X X h3 h4 h6 X X
h6 -> X X h3 h4 h5 X X
h7 -> h1 h2 X X X X h8
h8 -> h1 h2 X X X X h7
*** Results: 57% dropped (24/56 received)
mininet>
```

Figure 2: Ping luego de eliminar una conexión

Controlador antihorario

Posteriormente se creó otro controlador de manera que la tarea solo se pueda comunicar en sentido anti-horario

Red 2: Anillo con sentidos

Creación de la red

La segunda red se creó de la siguiente forma:

```
from mininet.topo import Topo
class MyTopo(Topo):
    def __init__(self):
        Topo.__init__(self)

        sw = []
        for i in range(4):
            sw.append(self.addSwitch(f's{i+1}', dpid = f'{i+1}'))

        hsts = []
        for i in range(5):
            hsts.append(self.addHost(f'h{i+1}',mac = f'10:00:00:00:00:0{i+1}'))

        for i in range(3):
            self.addLink(sw[i], sw[i-1], 3, 4)
        self.addLink(sw[2], sw[3], 3, 4)#conexion bidireccional

        for k in range(5):
            if k == 3:
                self.addLink(hsts[k-1], sw[k-1], 100, 2)
            elif  k == 4:
              self.addLink(hsts[k-1], sw[k-1], 100, 1)
            else:
              self.addLink(hsts[k-1], sw[k], 100, 1)


topos = {'MyTopo': (lambda: MyTopo())}
```

Controlador

Esta vez el controlador tiene muchas más especificaciones, pero se utilizó la misma base mencionada anteriormente.

Firewall

Finalmente se creó un firewall para la red:

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.revent import *
from pox.lib.addresses import EthAddr


rules = [['00:00:00:00:00:01','00:00:00:00:00:04'],#primer punto
        ['00:00:00:00:00:03','00:00:00:00:00:04'],
        ['00:00:00:00:00:05','00:00:00:00:00:04'],
        ['00:00:00:00:00:05','00:00:00:00:00:02'],
        ['00:00:00:00:00:03','00:00:00:00:00:02'],
        ['00:00:00:00:00:01','00:00:00:00:00:03'],##segundo punto
        ['00:00:00:00:00:02','00:00:00:00:00:03'],
        ['00:00:00:00:00:04','00:00:00:00:00:03'],
        ['00:00:00:00:00:04','00:00:00:00:00:05'],
        ['00:00:00:00:00:03','00:00:00:00:00:01'],#tercer punto
        ['00:00:00:00:00:04','00:00:00:00:00:01'],
        ]

class SDNFirewall (EventMixin):

    def __init__ (self):
        self.listenTo(core.openflow)
```

```
24
25      def _handle_ConnectionUp (self, event):
26          for rule in rules:
27              block = of.ofp_match()
28              block.dl_src = EthAddr(rule[0])
29              block.dl_dst = EthAddr(rule[1])
30              flow_mod = of.ofp_flow_mod()
31              flow_mod.match = block
32              event.connection.send(flow_mod)
33
34  def launch ():
35      core.registerNew(SDNFirewall)
```