Explicación Parte II: práctica SDN (topología y controlador)

Comandos a Utilizar Para Ejecutar los scripts:
-Comando para la topología (con fo siendo un número de 2 a 64): **sudo mn --custom scriptTopoParte1.py --topo mytopo,fo --controller remote –arp**
-Comando para el controlador: **ryu-manager skeleton.py**

Explicación de las partes de la Práctica:

- Explicación para la topología

```python
#!/usr/bin/python
# name: topo-2sw-2host.py


from mininet.topo import Topo
from mininet.node import OVSKernelSwitch, Host

class MyTopo( Topo ):
    "Simple topology example."

    def build( self,fo ):
        # Parámetro para el número de hosts por switch
        self.fo=fo

        # Nivel superior
        #top_switch = top_switch = self.addSwitch('Ts', cls=OVSKernelSwitch, dpid='0000000100002023')
        top_switch = self.addSwitch('Ts', cls=OVSKernelSwitch, dpid='0000000100002023')

        # Nivel de acceso
        for swNum in range(1, fo+1):
                Acceso = self.addSwitch('s{}'.format(swNum))
                self.addLink(top_switch, Acceso)
                # Nivel de hosts
                for hNum in range(1, 2*fo+1):
                    terminales = self.addHost('h_{}_{}'.format(swNum, hNum), cls=Host, ip='10.12.{}.{}/16'.format(swNum,hNum), defaul
                    self.addLink(terminales, Acceso)


topos = { 'mytopo': ( lambda fo=2: MyTopo(fo) ) }
```

  En función del valor de fo se crea una estructura de fo switches unidos a un top switch y a su vez 2*fo host por cada switch creado anteriormente (a excepción del top switch el cual se conecta unicamente a los fo switches).
  Fo es un número entre 2 y 64, y de no ser especificado el valor base es de 2.

Juan Navarlaz Urtubia

- Explicación para el controlador

```
for p in ev.msg.body:
    num_ports += 1

self.logger.info("\t Total number of ports of switch %s including control: %d" %
                 (dpid, num_ports))

num_ports = num_ports - 1 # Puertos fisicos -- quitamos el de control

#Determinar si es el switch Ts o los de acceso
if dpid < 65:
    #Obtener número de hosts y de switches
    nhosts = num_ports -1
    print(nhosts)
    ip_org='10.12.{}.0/24'.format(dpid)
    print(ip_org)
    self.add_flow_ip(datapath, 1, ip_org ,'10.12.0.0/16', 1)
    #Direcciones src
    for host in range(1, nhosts + 1):
        ip_org = '10.12.{}.{}/16'.format(dpid, host)
        ip_dest = '10.12.{}.{}/32'.format(dpid, host)
        self.add_flow_ip(datapath, 2, ip_org, ip_dest, host+1)

else:
    #Switch superior
    print("Switch sup " + str(dpid))
    #Direcciones src
    for switch in range(1, num_ports + 1):
        ip_org = '10.12.0.0/16'.format(switch)
        ip_dest = '10.12.{}.0/24'.format(switch)
        self.add_flow_ip(datapath, 1, ip_org, ip_dest, switch)
```

Dentro de la función “def port_desc_stats_reply_handler(self, ev)” se tratan los flow a añadir por cada switch, para ello por cada switch (de acceso) se añade una salida “default” al puerto uno (de prioridad menor a la de los host locales) en caso de que no haya un flow al host con mayor prioridad. Por cada host del switch se genera un flow desde cualquier mensaje de dirección que entre.
Lo mismo ocurre para el top switch (parte del else) el cual genera un flow para cada switch conectado.
Esto solo funcionaria para host con ip de esta red (10.12.0.0/16).