ARQUITECTURAS DE REDES AVANZADAS
PRÁCTICA 2
Balanceo de Carga con SDN
ENERO 2015

José Luis Cánovas Sánchez
joseluis.canovas2@um.es
48636907A

Resumen

Índice

1. Introducción

2. Topología mininet

La topología implementada en mininet corresponde, como se puede ver en la Figura 1 a dos switch conectados al controlador, 6 máquinas cliente y 4 servidores.

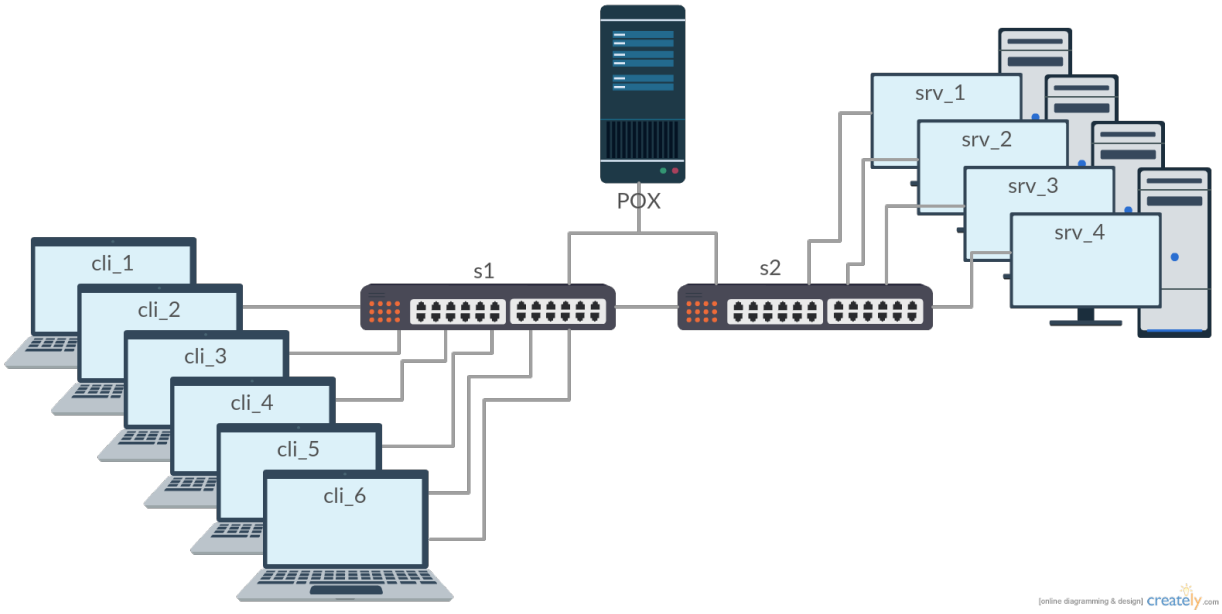

Figura 1: Topología

El código en python que define a la topología es el siguiente. Para facilitar las pruebas, a los servidores se les da una dirección MAC prefijada en el método *addHost*, y a las conexiones con el switch 's2' siempre se indican a qué puerto conectar, con el parámetro *port2* del método *addLink*.

```
# -*- coding: utf-8 -*-

from mininet.topo import Topo

class MyTopo (Topo):

    def __init__ (self):
        Topo.__init__( self )

        # Add switches
        sw_clients = self.addSwitch('s1')
        sw_servers = self.addSwitch('s2')

        # Add clients
        c1 = self.addHost('cli_1')
        c2 = self.addHost('cli_2')
        c3 = self.addHost('cli_3')
        c4 = self.addHost('cli_4')
        c5 = self.addHost('cli_5')
        c6 = self.addHost('cli_6')

        # Add servers
        s1 = self.addHost('srv_1', ip='10.0.0.101', mac='00:00:00:00:01:01')
        s2 = self.addHost('srv_2', ip='10.0.0.102', mac='00:00:00:00:01:02')
        s3 = self.addHost('srv_3', ip='10.0.0.103', mac='00:00:00:00:01:03')
```

```
        s4 = self.addHost('srv_4', ip='10.0.0.104', mac='00:00:00:00:01:04')

        # Add links
        self.addLink(sw_clients, sw_servers, port2=1)

        self.addLink(c1, sw_clients)
        self.addLink(c2, sw_clients)
        self.addLink(c3, sw_clients)
        self.addLink(c4, sw_clients)
        self.addLink(c5, sw_clients)
        self.addLink(c6, sw_clients)

        self.addLink(s1, sw_servers, port2=2)
        self.addLink(s2, sw_servers, port2=3)
        self.addLink(s3, sw_servers, port2=4)
        self.addLink(s4, sw_servers, port2=5)


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

3. Controlador POX con l2_learning

Realizamos un test básico de la topología anterior usando el fichero *l2_learning* que proporciona POX. La salida por pantalla de mininet durante la prueba es la siguiente:

```
$sudo mn --custom topo.py --topo mytopo --controller remote --test pingall
        *** Creating network
        *** Adding controller
        *** Adding hosts:
        cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
        *** Adding switches:
        s1 s2
        *** Adding links:
        (cli_1, s1) (cli_2, s1) (cli_3, s1) (cli_4, s1) (cli_5, s1)
         (cli_6, s1) (s1, s2) (srv_1, s2) (srv_2, s2) (srv_3, s2) (srv_4, s2)
        *** Configuring hosts
        cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
        *** Starting controller
        c0
        *** Starting 2 switches
        s1 s2 ...
        *** Waiting for switches to connect
        s1 s2
```

```
*** Ping: testing ping reachability
cli_1 -> cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_2 -> cli_1 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_3 -> cli_1 cli_2 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_4 -> cli_1 cli_2 cli_3 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_5 -> cli_1 cli_2 cli_3 cli_4 cli_6 srv_1 srv_2 srv_3 srv_4
cli_6 -> cli_1 cli_2 cli_3 cli_4 cli_5 srv_1 srv_2 srv_3 srv_4
srv_1 -> cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_2 srv_3 srv_4
srv_2 -> cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_3 srv_4
srv_3 -> cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_4
srv_4 -> cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3
*** Results: 0\% dropped (90/90 received)
*** Stopping 1 controllers
c0
*** Stopping 11 links
...........
*** Stopping 2 switches
s1 s2
*** Stopping 10 hosts
cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
*** Done
completed in 6.529 seconds
```

Todos los hosts (clientes y servidores) tienen conectividad entre ellos y ningún paquete ICMP se ha perdido. La topología de mininet es correcta y se conecta con el controlador POX.

4. Construcción del balanceador de carga

4.1. Módulo de balanceo en POX

Partimos del fichero *l2_learning.py* y añadimos las siguientes líneas de código en la clase LearningSwitch.

Las primeras líneas sirven para definir el método que por round-robin devuelve el siguiente puerto del switch s2 por el que balancear una nueva conexión a los servidores.

```
1  class LearningSwitch (object):
2      [...]
3      def __init__ (self, connection, transparent):
4          [...]
5          self.round_robin = 0
6          self.max_srvs = 4
7          self.frst_prt = 2
8
9      def roundRobin(self):
10         rr = (self.round_robin%self.max_srvs) + self.frst_prt
```

```
11          self.round_robin+=1
12          return rr
```

Estas líneas se añaden en _handle_PacketIn casi al inicio del método, después de aprender el puerto para la MAC del origen (añadiendo una entrada en *macToPort*), y comprobamos que sea un mensaje del switch 2, que es el que debe balancear, que es un ARP REQUEST y que pregunta por la máquina con IP la de nuestros servidores balanceados.

En caso de ser un paquete que cumple todo lo anterior, creamos un mensaje para el switch s2 que añadirá a su tabla un nuevo flujo para todo paquete con MAC origen la del cliente que se envíe por el puerto del switch que indique el método del round-robin.

```
1  def _handle_PacketIn (self, event):
2      [...]
3      # Round-Robin
4      if ( dpid_to_str(event.dpid) == "00-00-00-00-00-02" and
5                              packet.type == packet.ARP_TYPE and
6                          packet.payload.opcode == arp.REQUEST and
7                             packet.next.protodst == "10.0.0.101" ):
8          msg = of.ofp_flow_mod()
9          msg.match.dl_src = packet.src
10         msg.actions.append(of.ofp_action_output(port = self.roundRobin()))
11         #msg.idle_timeout = 10
12         #msg.hard_timeout = 30
13         msg.data = event.ofp
14         self.connection.send(msg)
15         return
16
17     if packet.dst.is_multicast:
18         flood() # 3a
19     [...]
```

4.2. Modificación topología

A la topología de mininet el único cambio que hay que aplicarle es modificar las líneas 22 a 26, modificando la IP en *addHost* por la 10.0.0.101, y que así sea única para todos los servidores.

4.3. Ejecución y prueba de PING

En el anexo al final de esta memoria se encuentra la salida por pantalla completa de las pruebas realizadas, y a continuación se muestra parte de las pruebas con ping aplicadas.

Para cada cliente se ejecuta la orden *cli_i ping -c 4 10.0.0.101*, 4 pings a la dirección de los servidores, que como vemos el primero tiene un retardo bastante considerable frente a los 3 siguientes, debido a que con el primero de todos el switch debe comunicarse con el controlador que se asigna el flujo para la MAC del cliente. Como el flujo se guarda en el switch, se aplica instantáneamente con los siguientes mensajes.

Además se realiza un *pingall* que muestra que los 6 clientes alcanzan, como en l2_learning a todas las máquinas. Sin embargo, los servidores, a pesar de poder hacer ping al resto de servidores, sólo reciben respuesta de uno o dos clientes, pues las respuestas tienen MAC origen la del cliente, y se reenvían al puerto asignado por round-robin en los pings anteriores. Por ejemplo, a los clientes 1 y 5 se les asignó, de los 4 servidores, el *srv_1*.

```
mininet> cli_6 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=38.9 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.448 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.597 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.352 ms

--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.352/10.082/38.933/16.657 ms
mininet> pingall
*** Ping: testing ping reachability
cli_1 -> cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_2 -> cli_1 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_3 -> cli_1 cli_2 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_4 -> cli_1 cli_2 cli_3 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_5 -> cli_1 cli_2 cli_3 cli_4 cli_6 srv_1 srv_2 srv_3 srv_4
cli_6 -> cli_1 cli_2 cli_3 cli_4 cli_5 srv_1 srv_2 srv_3 srv_4
srv_1 -> cli_1 X X X cli_5 X srv_2 srv_3 srv_4
srv_2 -> X cli_2 X X X cli_6 srv_1 srv_3 srv_4
srv_3 -> X X cli_3 X X X srv_1 srv_2 srv_4
srv_4 -> X X X cli_4 X X srv_1 srv_2 srv_3
*** Results: 20% dropped (72/90 received)
```

4.4. Tráfico y flujos

Flujos de los switches:

En el código del controlador de la sección anterior había comentadas dos líneas que indicaban al switch que la regla del nuevo flujo debería caducar. Para facilitar las pruebas, se comentan, y pasados unos segundos en los que las reglas de aprendizaje de macToPort sí caducan, con la orden en mininet *dpctl dump-flows* se muestran las tablas con los flujos asignados por round-robin.

```
mininet> dpctl dump-flows
```

```
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=390.693s, table=0, n_packets=24, n_bytes=1512, idle_age=309,
dl_src=fe:1c:e8:bb:47:2b actions=output:4
cookie=0x0, duration=397.979s, table=0, n_packets=25, n_bytes=1554, idle_age=312,
dl_src=5a:fb:3a:24:c3:d0 actions=output:3
cookie=0x0, duration=368.291s, table=0, n_packets=21, n_bytes=1386, idle_age=303,
dl_src=26:1a:ac:72:e4:51 actions=output:3
cookie=0x0, duration=383.206s, table=0, n_packets=23, n_bytes=1470, idle_age=303,
dl_src=ba:d3:b1:08:9d:20 actions=output:5
cookie=0x0, duration=406.713s, table=0, n_packets=26, n_bytes=1596, idle_age=315,
dl_src=86:bd:bc:16:52:e3 actions=output:2
cookie=0x0, duration=376.218s, table=0, n_packets=21, n_bytes=1386, idle_age=306,
dl_src=ca:6d:f2:8c:c1:a9 actions=output:2
```

Se observa que todas pertenecen a la tabla 0, y que en *dl_src* se indica la MAC de un cliente y en *actions=output* el puerto por donde reenviar el paquete. Usando la información de los *ifconfig* que se encuentra en el anexo, se puede hacer la correspondencia de cada cliente con su MAC y por tanto su flujo.

A los clientes 1 y 5 les corresponde el puerto 2, es decir, el servidor 1. A los clientes 2 y 6 el puerto 3, servidor 2. Al cliente 3 el puerto 4, servidor 3. Y al cliente 4 el puerto 5, servidor 4.

Las tablas de flujos de los switchs sin que hayan caducado las entradas de macToPort son las siguientes:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=10.672s, table=0, n_packets=3, n_bytes=126, idle_timeout=10,
 hard_timeout=30, idle_age=8, priority=65535,arp,in_port=7,vlan_tci=0x0000,
  dl_src=02:3f:e7:62:af:f1,dl_dst=00:00:00:00:01:04,arp_spa=10.0.0.6,
  arp_tpa=10.0.0.101,arp_op=2 actions=output:1
cookie=0x0, duration=8.627s, table=0, n_packets=1, n_bytes=42, idle_timeout=10,
 hard_timeout=30, idle_age=8, priority=65535,arp,in_port=5,vlan_tci=0x0000,
  dl_src=0a:27:a9:ce:34:da,dl_dst=00:00:00:00:01:04,arp_spa=10.0.0.4,
  arp_tpa=10.0.0.101,arp_op=2 actions=output:1
cookie=0x0, duration=8.666s, table=0, n_packets=1, n_bytes=42, idle_timeout=10,
 hard_timeout=30, idle_age=8, priority=65535,arp,in_port=1,vlan_tci=0x0000,
  dl_src=00:00:00:00:01:04,dl_dst=0a:27:a9:ce:34:da,arp_spa=10.0.0.101,
  arp_tpa=10.0.0.4,arp_op=1 actions=output:5
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
```

```
cookie=0x0, duration=8.673s, table=0, n_packets=1, n_bytes=42, idle_timeout=10,
 hard_timeout=30, idle_age=8, priority=65535,arp,in_port=5,vlan_tci=0x0000,dl_src=00:00:00:00:01:04,dl_
cookie=0x0, duration=117.856s, table=0, n_packets=26, n_bytes=1596, idle_age=20,
 dl_src=ba:b9:7f:25:67:db actions=output:2
cookie=0x0, duration=110.578s, table=0, n_packets=25, n_bytes=1554, idle_age=17,
 dl_src=d6:a7:d3:3c:41:ed actions=output:3
cookie=0x0, duration=78.586s, table=0, n_packets=21, n_bytes=1386, idle_age=8,
 dl_src=02:3f:e7:62:af:f1 actions=output:3
cookie=0x0, duration=86.319s, table=0, n_packets=21, n_bytes=1386, idle_age=11,
 dl_src=e6:aa:ea:b9:70:6a actions=output:2
cookie=0x0, duration=103.169s, table=0, n_packets=24, n_bytes=1512, idle_age=14,
 dl_src=f6:46:72:d8:ae:cc actions=output:4
cookie=0x0, duration=93.593s, table=0, n_packets=23, n_bytes=1470, idle_age=8,
 dl_src=0a:27:a9:ce:34:da actions=output:5
```

A. Test ping

```
mininet@mininet-vm:~/GitHub/mininet/scripts$ sudo ./testWcont topo.py
*** Creating network
*** Adding controller
*** Adding hosts:
cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
*** Adding switches:
s1 s2
*** Adding links:
(cli_1, s1) (cli_2, s1) (cli_3, s1) (cli_4, s1) (cli_5, s1) (cli_6, s1)
 (s1, s2) (srv_1, s2) (srv_2, s2) (srv_3, s2) (srv_4, s2)
*** Configuring hosts
cli_1 cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> cli_1 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=12.2 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=1.21 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.063 ms
```

```
--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.063/3.441/12.288/5.127 ms
mininet> cli_2 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=31.9 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.810 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.776 ms

--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.071/8.392/31.914/13.583 ms
mininet> cli_3 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=40.4 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=1.28 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.671 ms

--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.072/10.614/40.426/17.217 ms
mininet> cli_4 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=28.5 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.407 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.083 ms

--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.073/7.284/28.574/12.292 ms
mininet> cli_5 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=19.3 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.447 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.846 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.087 ms

--- 10.0.0.101 ping statistics ---
```

```
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.087/5.178/19.332/8.176 ms
mininet> cli_6 ping -c 4 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=38.9 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.448 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.597 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.352 ms

--- 10.0.0.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.352/10.082/38.933/16.657 ms
mininet> pingall
*** Ping: testing ping reachability
cli_1 -> cli_2 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_2 -> cli_1 cli_3 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_3 -> cli_1 cli_2 cli_4 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_4 -> cli_1 cli_2 cli_3 cli_5 cli_6 srv_1 srv_2 srv_3 srv_4
cli_5 -> cli_1 cli_2 cli_3 cli_4 cli_6 srv_1 srv_2 srv_3 srv_4
cli_6 -> cli_1 cli_2 cli_3 cli_4 cli_5 srv_1 srv_2 srv_3 srv_4
srv_1 -> cli_1 X X X cli_5 X srv_2 srv_3 srv_4
srv_2 -> X cli_2 X X X cli_6 srv_1 srv_3 srv_4
srv_3 -> X X cli_3 X X X srv_1 srv_2 srv_4
srv_4 -> X X X cli_4 X X srv_1 srv_2 srv_3
*** Results: 20% dropped (72/90 received)
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
*** s2 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=390.693s, table=0, n_packets=24, n_bytes=1512, idle_age=309,
  dl_src=fe:1c:e8:bb:47:2b actions=output:4
 cookie=0x0, duration=397.979s, table=0, n_packets=25, n_bytes=1554, idle_age=312,
  dl_src=5a:fb:3a:24:c3:d0 actions=output:3
 cookie=0x0, duration=368.291s, table=0, n_packets=21, n_bytes=1386, idle_age=303,
  dl_src=26:1a:ac:72:e4:51 actions=output:3
 cookie=0x0, duration=383.206s, table=0, n_packets=23, n_bytes=1470, idle_age=303,
  dl_src=ba:d3:b1:08:9d:20 actions=output:5
 cookie=0x0, duration=406.713s, table=0, n_packets=26, n_bytes=1596, idle_age=315,
  dl_src=86:bd:bc:16:52:e3 actions=output:2
 cookie=0x0, duration=376.218s, table=0, n_packets=21, n_bytes=1386, idle_age=306,
```

```
  dl_src=ca:6d:f2:8c:c1:a9 actions=output:2
mininet> cli_1 ifconfig
cli_1-eth0 Link encap:Ethernet  HWaddr 86:bd:bc:16:52:e3
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5306 (5.3 KB)  TX bytes:2786 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> cli_2 ifconfig
cli_2-eth0 Link encap:Ethernet  HWaddr 5a:fb:3a:24:c3:d0
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5306 (5.3 KB)  TX bytes:2786 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> cli_3 ifconfig
cli_3-eth0 Link encap:Ethernet  HWaddr fe:1c:e8:bb:47:2b
          inet addr:10.0.0.3  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:5306 (5.3 KB)  TX bytes:2786 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> cli_4 ifconfig
cli_4-eth0 Link encap:Ethernet  HWaddr ba:d3:b1:08:9d:20
          inet addr:10.0.0.4  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5306 (5.3 KB)  TX bytes:2786 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> cli_5 ifconfig
cli_5-eth0 Link encap:Ethernet  HWaddr ca:6d:f2:8c:c1:a9
          inet addr:10.0.0.5  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5264 (5.2 KB)  TX bytes:2744 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> cli_6 ifconfig
cli_6-eth0 Link encap:Ethernet  HWaddr 26:1a:ac:72:e4:51
          inet addr:10.0.0.6  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5306 (5.3 KB)  TX bytes:2786 (2.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

Referencias