MININET 4: TRABAJO DE ADMINISTRACIÓN REDES

Autor:Alejandro Rodríguez Rojas

Indice

1 Introducción

Queremos hacer una conexión mediante una máquina virtual a mininet, y configurar sus host usando el programa Xterm, se nos pide configurar la IP,poder hacer ping entre las máquinas y las tablas de enrutamiento.

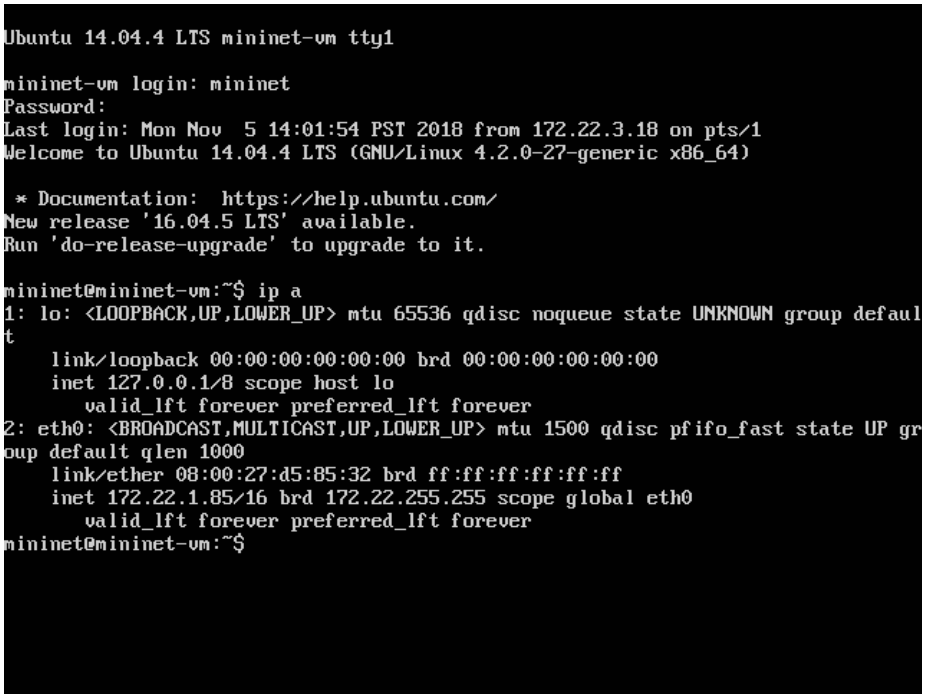
2 Conexión a Mininet

Iniciamos VirtualBox y nos descargamos la máquina virtual mediante la URL:

http://mininet.org/download/

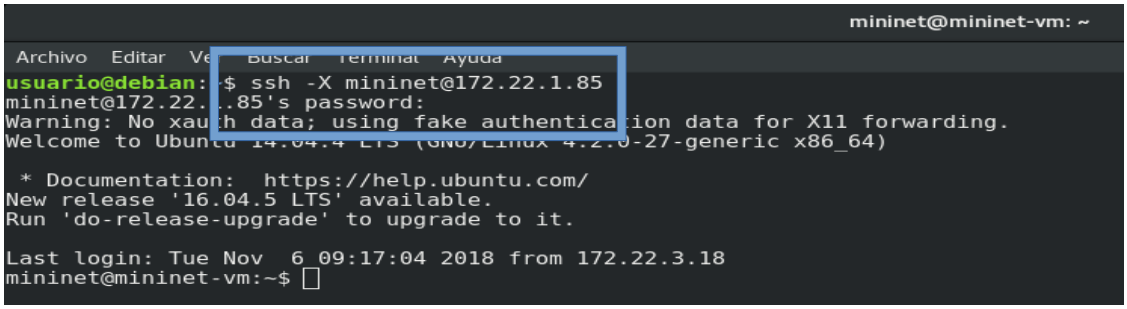
Acto seguido ejecutamos la máquina Virtual en VirtualBox.

Te pide usuario y contraseña(ambas son mininet).



Acto seguido ponemos la máquina en modo Adaptador puente y nos conectamos mediante SSH (utilizaremos el parámetro -X para conectarnos a Xterm)

a la máquina virtual.

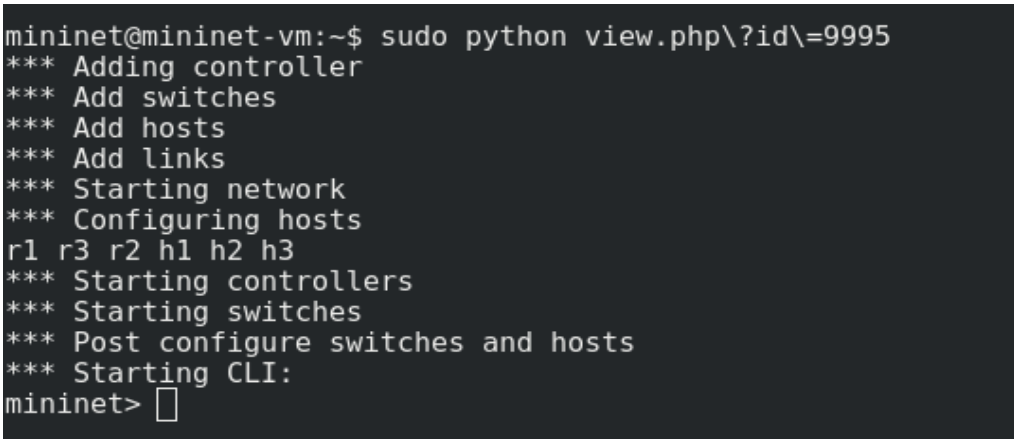


3 Conexión al entorno

Ya estamos conectados mediante SSH a la máquina Virtual.

Debemos abrir el prográma Python proporcionado por el profresor para que se nos ejecute el entorno donde vamos a trabajar:

sudo python {nombre}



4 Conexión mediante Xterm

Al entrar al entorno, utilizaremos el siguiente comando para conectarnos a los dos hosts y al router, que son con los que debemos trabajar:

```
xterm h1 h2 h3 r1 r2 r3
```

Siendo:

h1→Host 1

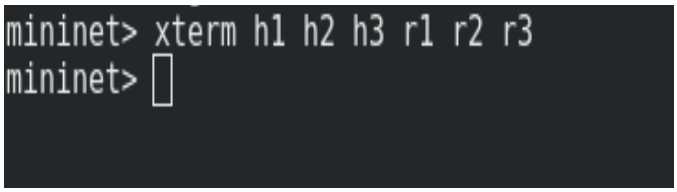
h2 → Host 2

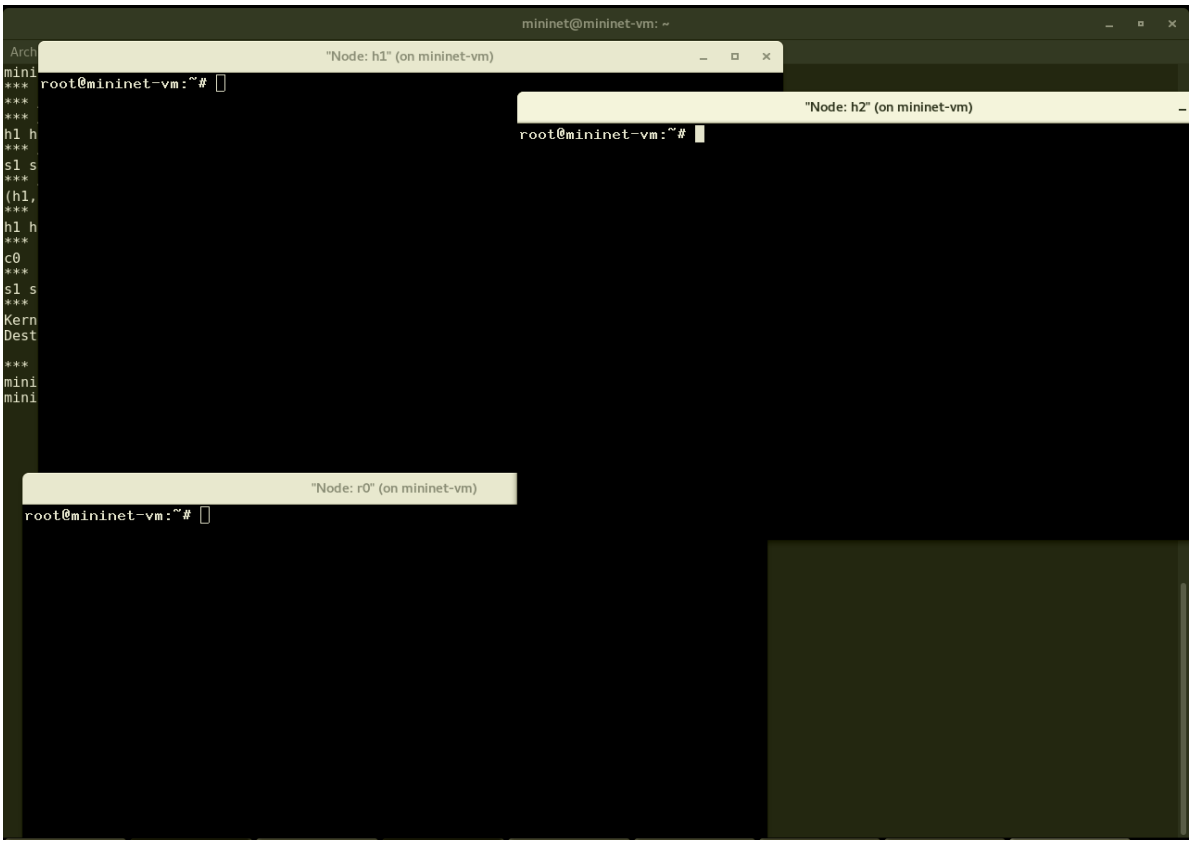
h3 → Host 3

r1 → Router 1

r2 → Router 2

r3 → Router 3





5 Cambio de IP de los Host y del Router

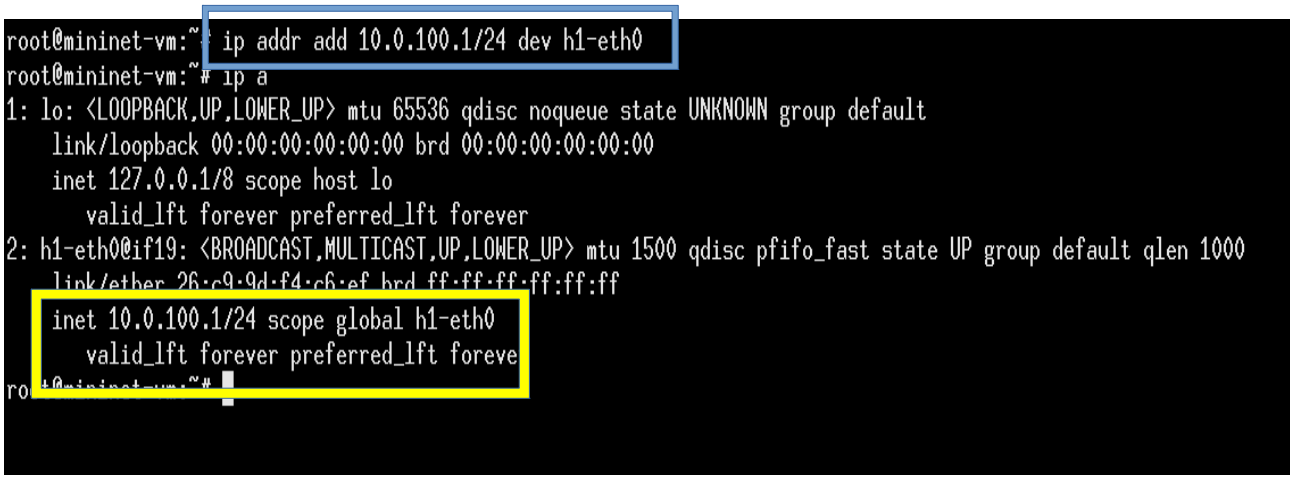
5.1 Método 1

Los host y el Router en un principio no tendrían IP, por lo que debemos darle la IP asignada por el profesor:(192.168.0.0/24 para host1 y 172.32.0.0/12 para host2 10.1.0.0/7 para host3)

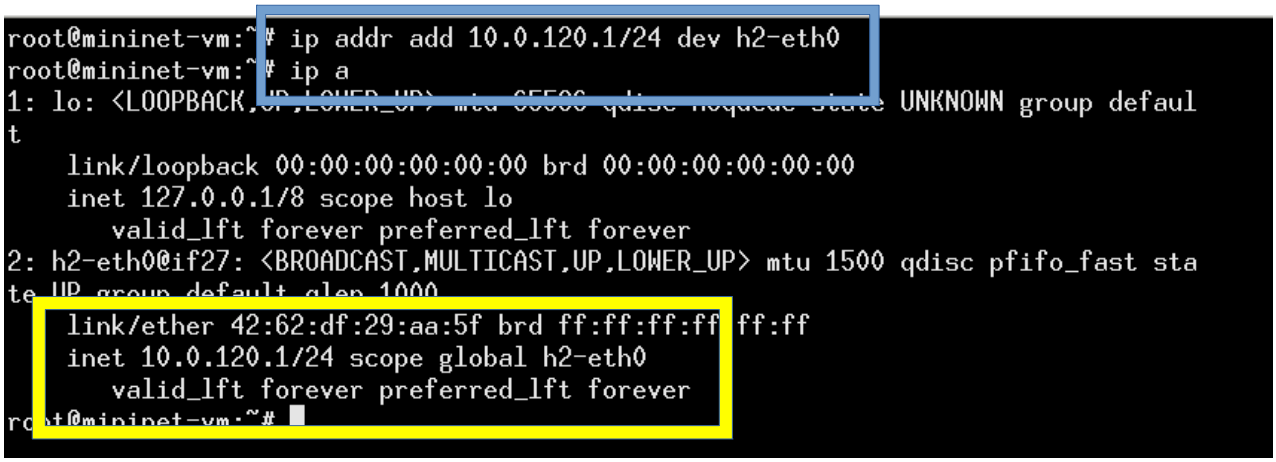
Deberemos utilizar un comando para cambiar la IP de los host y del router:

ip addr add {IP} dev {Nombre de la tarjeta}

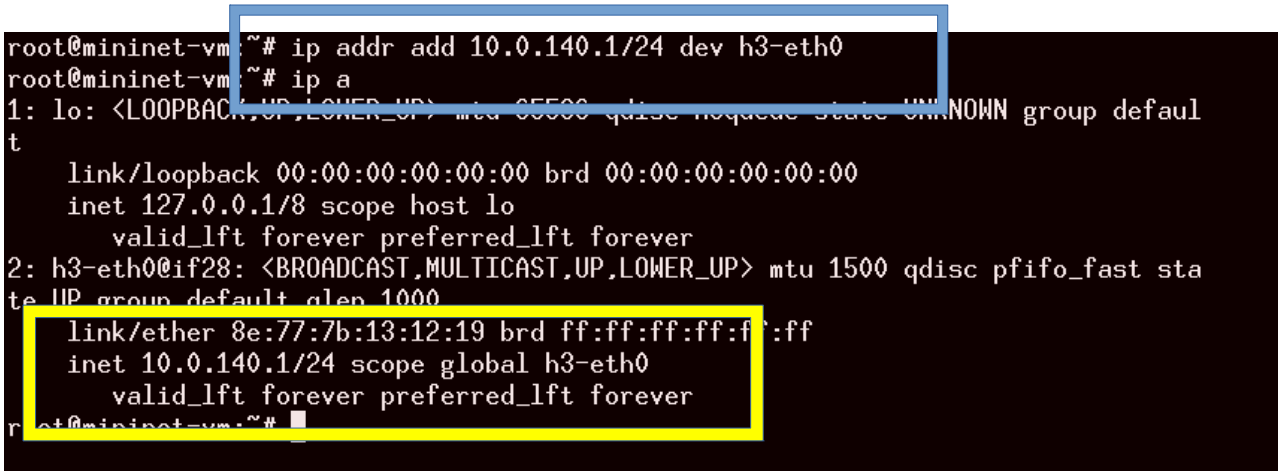
H1→


root@mininet-vm:~# ip addr add 10.0.100.1/24 dev h1-eth0
root@mininet-vm:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: h1-eth0@if19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
inet 10.0.100.1/24 scope global h1-eth0
valid_lft forever preferred_lft forever

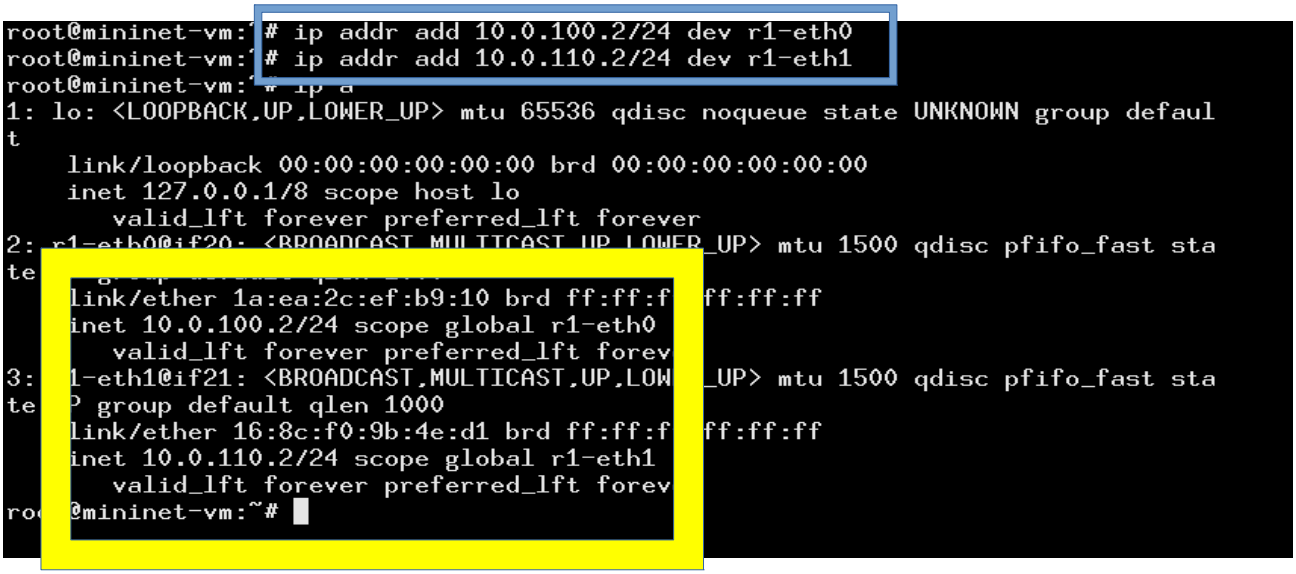
H2→


root@mininet-vm:~# ip addr add 10.0.120.1/24 dev h2-eth0
root@mininet-vm:~# ip a
UNKNOWN group default
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: h2-eth0@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
link/ether 42:62:df:29:aa:5f brd ff:ff:ff:ff:ff:ff
inet 10.0.120.1/24 scope global h2-eth0
valid_lft forever preferred_lft forever

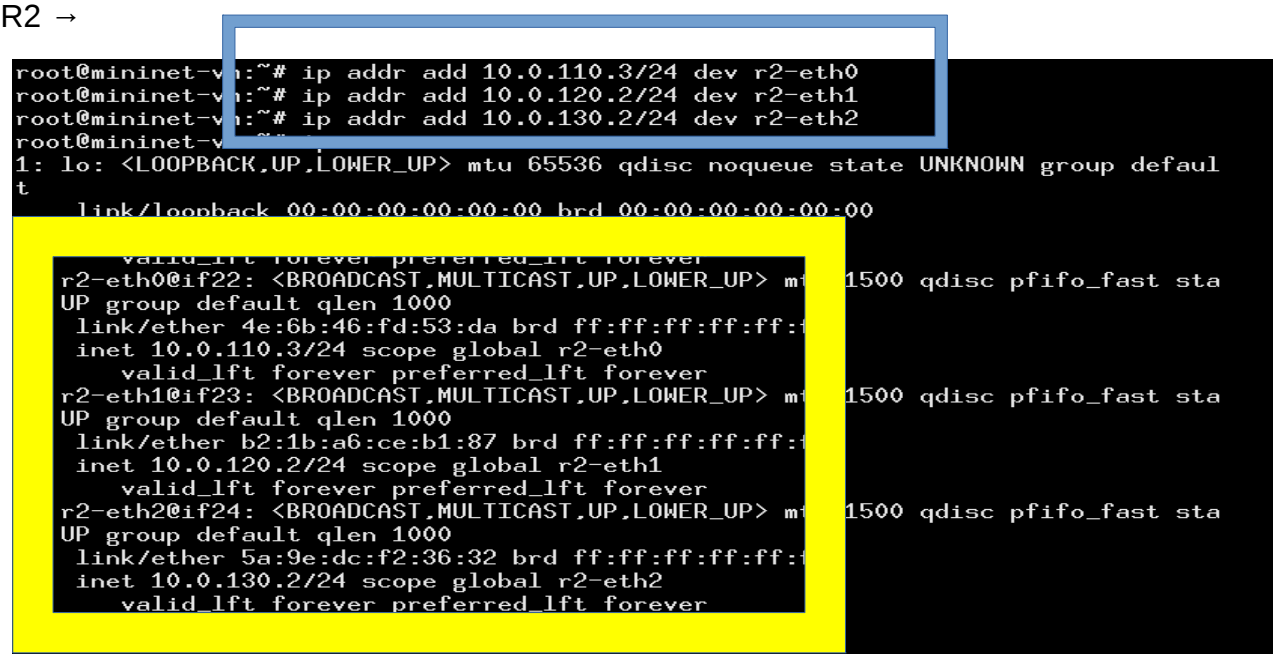
H3 →


root@mininet-vm:~# ip addr add 10.0.140.1/24 dev h3-eth0
root@mininet-vm:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: h3-eth0@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te UP group default qlen 1000
link/ether 8e:77:7b:13:12:19 brd ff:ff:ff:ff:ff:ff
inet 10.0.140.1/24 scope global h3-eth0
valid_lft forever preferred_lft forever

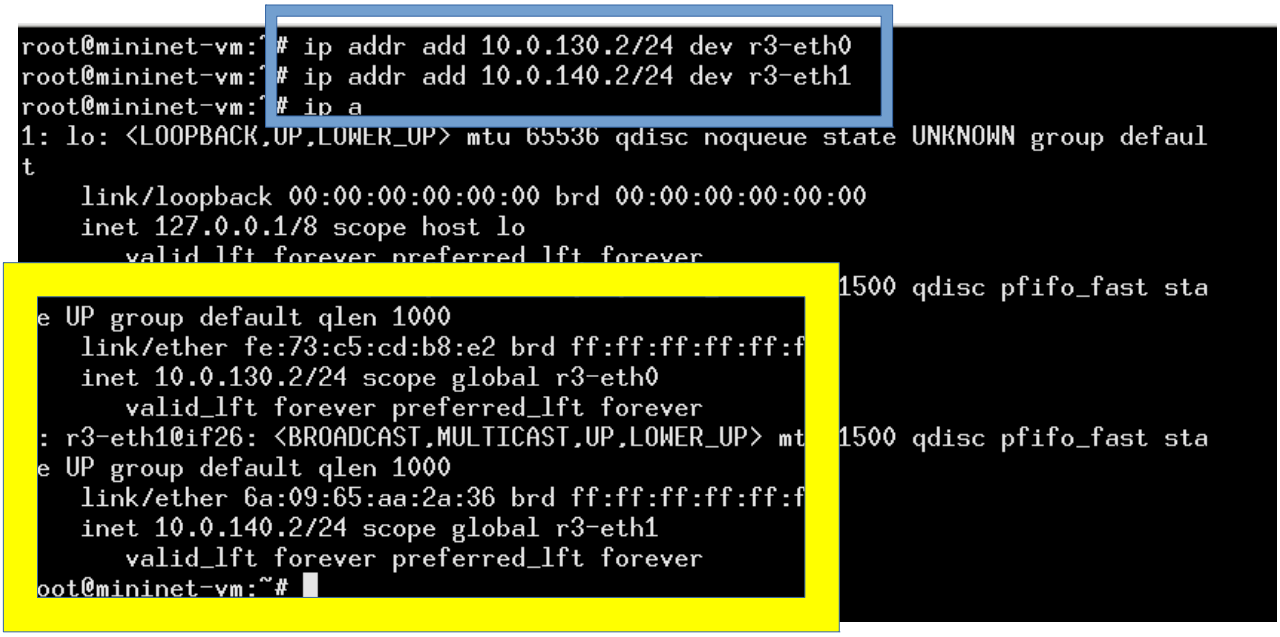
R1→


root@mininet-vm:~# ip addr add 10.0.100.2/24 dev r1-eth0
root@mininet-vm:~# ip addr add 10.0.110.2/24 dev r1-eth1
root@mininet-vm:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: r1-eth0@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
te
link/ether 1a:ea:2c:ef:b9:10 brd ff:ff:f ff:ff:ff
inet 10.0.100.2/24 scope global r1-eth0
valid_lft forever preferred_lft forev
3: 1-eth1@if21: <BROADCAST,MULTICAST,UP,LOW _UP> mtu 1500 qdisc pfifo_fast sta
te P group default qlen 1000
link/ether 16:8c:f0:9b:4e:d1 brd ff:ff:f ff:ff:ff
inet 10.0.110.2/24 scope global r1-eth1
valid_lft forever preferred_lft forev
ro @mininet-vm:~#

R2 →


root@mininet-vm:~# ip addr add 10.0.110.3/24 dev r2-eth0
root@mininet-vm:~# ip addr add 10.0.120.2/24 dev r2-eth1
root@mininet-vm:~# ip addr add 10.0.130.2/24 dev r2-eth2
root@mininet-vm:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
valid_lft forever preferred_lft forever
r2-eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mt 1500 qdisc pfifo_fast sta
UP group default qlen 1000
link/ether 4e:6b:46:fd:53:da brd ff:ff:ff:ff:ff:f
inet 10.0.110.3/24 scope global r2-eth0
valid_lft forever preferred_lft forever
r2-eth1@if23: <BROADCAST,MULTICAST,UP,LOWER_UP> mt 1500 qdisc pfifo_fast sta
UP group default qlen 1000
link/ether b2:1b:a6:ce:b1:87 brd ff:ff:ff:ff:ff:f
inet 10.0.120.2/24 scope global r2-eth1
valid_lft forever preferred_lft forever
r2-eth2@if24: <BROADCAST,MULTICAST,UP,LOWER_UP> mt 1500 qdisc pfifo_fast sta
UP group default qlen 1000
link/ether 5a:9e:dc:f2:36:32 brd ff:ff:ff:ff:ff:f
inet 10.0.130.2/24 scope global r2-eth2
valid_lft forever preferred_lft forever

R3 →



5.2 Método 2

Abrimos miniedit con el comando:

```
sudo ~/mininet/examples/miniedit.py
```

Y hacemos el esquema del punto 8.

Luego lo guardamos y le damos a Export level 2 script.

Abrimos el archivo con el comando:

```
sudo nano {NombreArchivo}
```

```
  GNU nano 2.2.6                          File: escenario4.py

#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/24')

    info( '*** Adding controller\n' )
    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch, failMode='standalone')
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch, failMode='standalone')
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch, failMode='standalone')
    r1 = net.addHost('r1', cls=Node, ip='10.0.100.2/24')
    r1.cmd('sysctl -w net.ipv4.ip_forward=1')
    r3 = net.addHost('r3', cls=Node, ip='10.0.140.2/24 ')
    r3.cmd('sysctl -w net.ipv4.ip_forward=1')
    r2 = net.addHost('r2', cls=Node, ip='10.0.120.2/24')
    r2.cmd('sysctl -w net.ipv4.ip_forward=1')
    s5 = net.addSwitch('s5', cls=OVSKernelSwitch, failMode='standalone')
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch, failMode='standalone')

    info( '*** Add hosts\n')
    h1 = net.addHost('h1', cls=Host, ip='10.0.100.1/24', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.120.1/24', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.140.1/24', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(h1, s1)
    net.addLink(s1, r1)
    net.addLink(r1, s2)
    net.addLink(s2, r2)
    net.addLink(r2, s3)
    net.addLink(s3, h2)
                                                [ Read 71 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

En ese archivo modificamos las IP que queramos utilizar.

6 Modificación del Enrutamiento

Para modificar las tablas de enrutamiento debemos usar el comando:

ip r add {IPDestino} via {IPGateway}

h1→

```
root@mininet-vm:~# ip r add default via 10.0.100.2
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.100.2      0.0.0.0         UG        0 0          0 h1-eth0
10.0.100.0      *               255.255.255.0   U         0 0          0 h1-eth0
root@mininet-vm:~#
```

h2→

```
root@mininet-vm:~# ip r add default via 10.0.120.2
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.120.2      0.0.0.0         UG        0 0          0 h2-eth0
10.0.120.0      *               255.255.255.0   U         0 0          0 h2-eth0
root@mininet-vm:~#
```

h3 →

```
root@mininet-vm:~# ip r add default via 10.0.140.2
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.140.2      0.0.0.0         UG        0 0          0 h3-eth0
10.0.140.0      *               255.255.255.0   U         0 0          0 h3-eth0
root@mininet-vm:~#
```

r1 →

```
root@mininet-vm:~# ip r add default via 10.0.110.3
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.110.3      0.0.0.0         UG        0 0          0 r1-eth1
10.0.100.0      *               255.255.255.0   U         0 0          0 r1-eth0
10.0.110.0      *               255.255.255.0   U         0 0          0 r1-eth1
root@mininet-vm:~#
```

r2 →

```
root@mininet-vm:~# ip r add default via 10.0.110.2
root@mininet-vm:~# ip r add 10.0.140.2 via 10.0.130.3
root@mininet-vm:~# ip r add 10.0.140.1 via 10.0.130.3
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.110.2      0.0.0.0         UG        0 0          0 r2-eth0
10.0.110.0      *               255.255.255.0   U         0 0          0 r2-eth0
10.0.120.0      *               255.255.255.0   U         0 0          0 r2-eth1
10.0.130.0      *               255.255.255.0   U         0 0          0 r2-eth2
10.0.140.1      10.0.130.3      255.255.255.255 UGH       0 0          0 r2-eth2
10.0.140.2      10.0.130.3      255.255.255.255 UGH       0 0          0 r2-eth2
root@mininet-vm:~#
```

r3 →

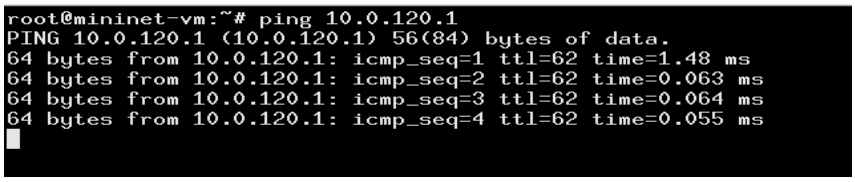
```
root@mininet-vm:~# ip r add default via 10.0.130.2
root@mininet-vm:~# route -e
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.130.2      0.0.0.0         UG        0 0          0 r3-eth0
10.0.130.0      *               255.255.255.0   U         0 0          0 r3-eth0
10.0.140.0      *               255.255.255.0   U         0 0          0 r3-eth1
root@mininet-vm:~#
```

7 Ping hacia los Hosts y Router

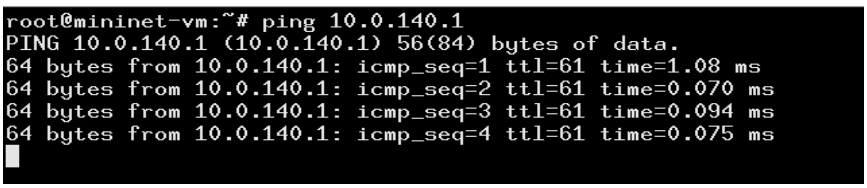
Para comprobar que todo esta correctamente realizamos los pings de host1 a host2 y de host1 al router y host2 al router, de host 1 a host 3, de host 2 a host 3 y de host 3 a router usando el comando:

ping {IP}

H1 a H2→



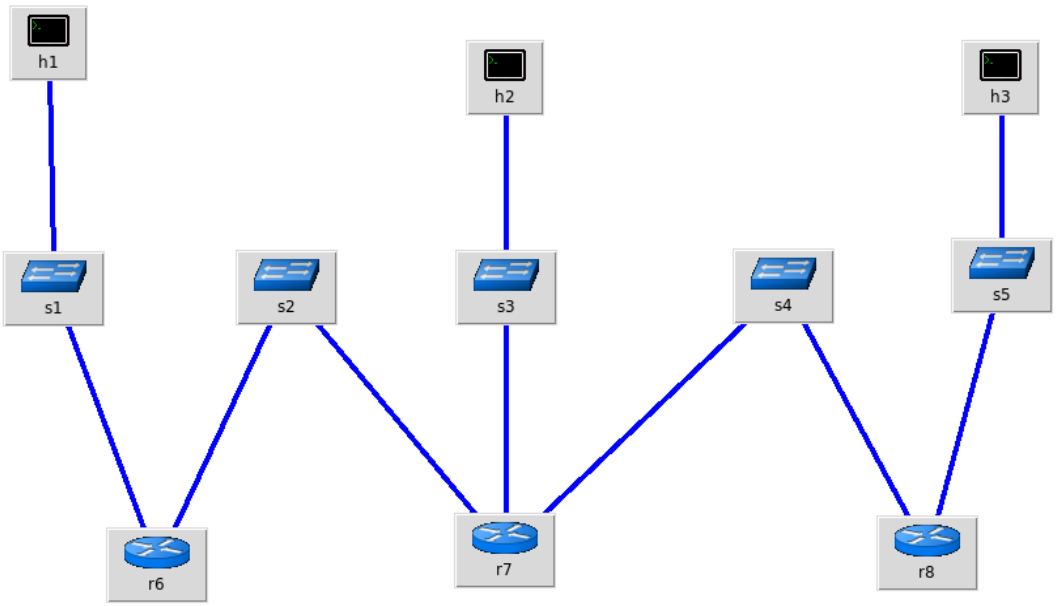
H1 a H3 →



H2 a H3 →

```
root@mininet-vm:~# ping 10.0.140.1
PING 10.0.140.1 (10.0.140.1) 56(84) bytes of data.
64 bytes from 10.0.140.1: icmp_seq=1 ttl=62 time=0.814 ms
64 bytes from 10.0.140.1: icmp_seq=2 ttl=62 time=0.061 ms
64 bytes from 10.0.140.1: icmp_seq=3 ttl=62 time=0.056 ms
64 bytes from 10.0.140.1: icmp_seq=4 ttl=62 time=0.047 ms
```

8 Esquema del problema



9 Captura Tráfico

Para capturar el tráfico acordado debemos usar el siguiente comando:

tcpdump -vi {Interfaz}

```
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 372, length 64
10:55:53.473990 IP (tos 0x0, ttl 62, id 11314, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 373, length 64
10:55:53.474032 IP (tos 0x0, ttl 63, id 36024, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 373, length 64
10:55:54.473869 IP (tos 0x0, ttl 62, id 11390, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 374, length 64
10:55:54.473896 IP (tos 0x0, ttl 63, id 36205, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 374, length 64
10:55:55.473974 IP (tos 0x0, ttl 62, id 11582, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 375, length 64
10:55:55.474014 IP (tos 0x0, ttl 63, id 36347, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 375, length 64
10:55:56.473922 IP (tos 0x0, ttl 62, id 11720, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 376, length 64
10:55:56.473949 IP (tos 0x0, ttl 63, id 36509, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 376, length 64
10:55:57.473934 IP (tos 0x0, ttl 62, id 11948, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 377, length 64
10:55:57.473973 IP (tos 0x0, ttl 63, id 36754, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 377, length 64
10:55:58.473908 IP (tos 0x0, ttl 62, id 12189, offset 0, flags [DF], proto ICMP
(1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 378, length 64
10:55:58.473933 IP (tos 0x0, ttl 63, id 37002, offset 0, flags [none], proto ICM
P (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 378, length 64
10:55:59.473912 IP (tos 0x0, ttl 62, id 12254, offset 0, flags [DF], proto ICMP (1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 379, length 64
10:55:59.473936 IP (tos 0x0, ttl 63, id 37206, offset 0, flags [none], proto ICMP (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 379, length 64
10:56:00.473762 IP (tos 0x0, ttl 62, id 12321, offset 0, flags [DF], proto ICMP (1), length 84)
    10.0.100.1 > 10.0.140.1: ICMP echo request, id 3115, seq 380, length 64
10:56:00.473796 IP (tos 0x0, ttl 63, id 37260, offset 0, flags [none], proto ICMP (1), length 84)
    10.0.140.1 > 10.0.100.1: ICMP echo reply, id 3115, seq 380, length 64
```

10 Conclusión

Con este ejercicio hemos aprendido a usar Mininet, junto a la configuración de una red y poder hacer ping entre dos host de esa misma red y configurar el enrutamiento de esta red creada.