Redes Software - Laboratorio

Introducción a SDN

Grupo 2
100451366 - Noel Andolz Aguado
100429717 - David Peño Gutiérrez

Hito 1:

1.1 Introducción:

1.1.1:

Instalamos y activamos el controlador OpenFlow.
Utilizamos el paquete *ovs-testcontroller* y le damos el nombre *ovs-controller*.

```
student@rs-2025:~$ sudo apt install openvswitch-testcontroller
[sudo] password for student:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  openvswitch-pki
The following NEW packages will be installed:
  openvswitch-pki openvswitch-testcontroller
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 286 kB of archives.
After this operation, 943 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openvswitch-pki all 2.17.9-0ubuntu0.22.04.1 [2,580 B]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openvswitch-testcontroller amd64 2.17.9-0ubuntu0.22.04.1 [284 kB]
Fetched 286 kB in 1s (413 kB/s)
Selecting previously unselected package openvswitch-pki.
(Reading database ... 231577 files and directories currently installed.)
Preparing to unpack .../openvswitch-pki_2.17.9-0ubuntu0.22.04.1_all.deb ...
Unpacking openvswitch-pki (2.17.9-0ubuntu0.22.04.1) ...
Selecting previously unselected package openvswitch-testcontroller.
Preparing to unpack .../openvswitch-testcontroller_2.17.9-0ubuntu0.22.04.1_amd64
.deb ...
Unpacking openvswitch-testcontroller (2.17.9-0ubuntu0.22.04.1) ...
Setting up openvswitch-pki (2.17.9-0ubuntu0.22.04.1) ...
Creating controllerca...
Creating switchca...
Setting up openvswitch-testcontroller (2.17.9-0ubuntu0.22.04.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
student@rs-2025:~$ sudo services openvswitch-testcontroller force-stop
sudo: services: command not found
student@rs-2025:~$ sudo service openvswitch-testcontroller force-stop
student@rs-2025:~$ cd /usr/bin
student@rs-2025:/usr/bin$ sudo ln -sf ovs-testcontroller ovs-controller
student@rs-2025:/usr/bin$
```

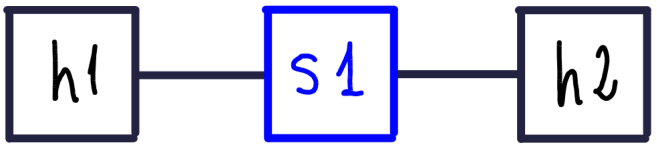
1.1.2:

Arrancamos mininet con el comando *sudo mn*. Esto inicia una topología por defecto con los siguientes elementos:

- 1 switch
- 2 hosts
- 1 controlador

```
student@rs-2025:/usr/bin$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Podemos ver que tanto el *h1* como el *h2* están conectados al *switch1*.



Tal y como se nos indica en la práctica, probamos a ejecutar algunos comandos como *pingall*

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Vemos que el ping se completa para los dos equipos finales (*h1* y *h2*).

El siguiente comando de ejemplo a ejecutar nos permite verificar la conectividad entre los dispositivos enviando 5 paquetes ICMP desde *h1* hasta *h2*:

```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.50 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.22 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.148 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.090 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4064ms
rtt min/avg/max/mdev = 0.090/1.012/3.495/1.312 ms
mininet>
```

Podemos ver que han llegado los 5 paquetes, por lo que hay una pérdida de 0%, y el tiempo que le ha tomado a cada uno de ellos.
Ahora probamos a ver la dirección IP de *h1*, para ello usaremos el comando *h1 ip route*:

```
mininet> h1 ip route
10.0.0.0/8 dev h1-eth0 proto kernel scope link src 10.0.0.1
mininet>
```

En este caso, la dirección IP de la interfaz *h1-eth0* es 10.0.0.1
De este comando también podemos saber que la red a la que está conectada el host *h1* es 10.0.0.0/8
Salimos de mininet usando el comando *quit* y limpiamos el entorno con *sudo mn -c*

```
student@rs-2025:/usr/bin$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
student@rs-2025:/usr/bin$
```

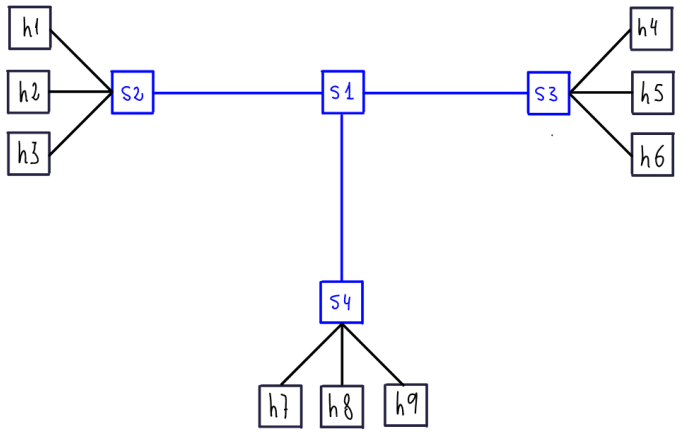
1.2 Controlando el comportamiento de Mininet desde la línea de comando:

1.2.1:

Probamos la siguiente topología *sudo mn –topo tree2,3*

```
student@rs-2025:/usr/bin$ sudo mn --topo tree,2,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3, h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet>
```

La topología va a ser de árbol. Va a tener $3^2 = 9$ hosts.
Su estructura va a ser la siguiente:



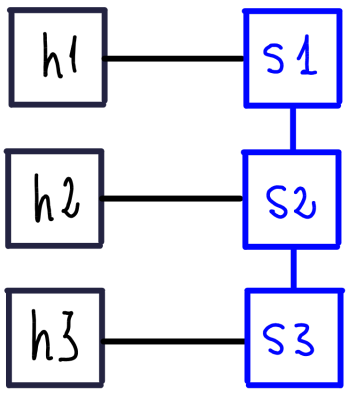
¿CUÁL ES LA DIFERENCIA?
La principal diferencia es que con *sudo mn* solo tenemos 2 hosts y 1 switch, mientras que con *sudo mn - -topo tree,2,3* tenemos 9 hosts y 4 switches.

1.2.2:

Ejecutamos mininet con una nueva topología: *sudo mn - -topo linear,3*:

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Podemos ver que presenta 3 hosts y 3 switches con la siguiente estructura:



Y vemos cuáles son las direcciones MAC de los equipo finales con los comandos: *h1 ip addr show*

h2 ip addr show

h3 ip addr show

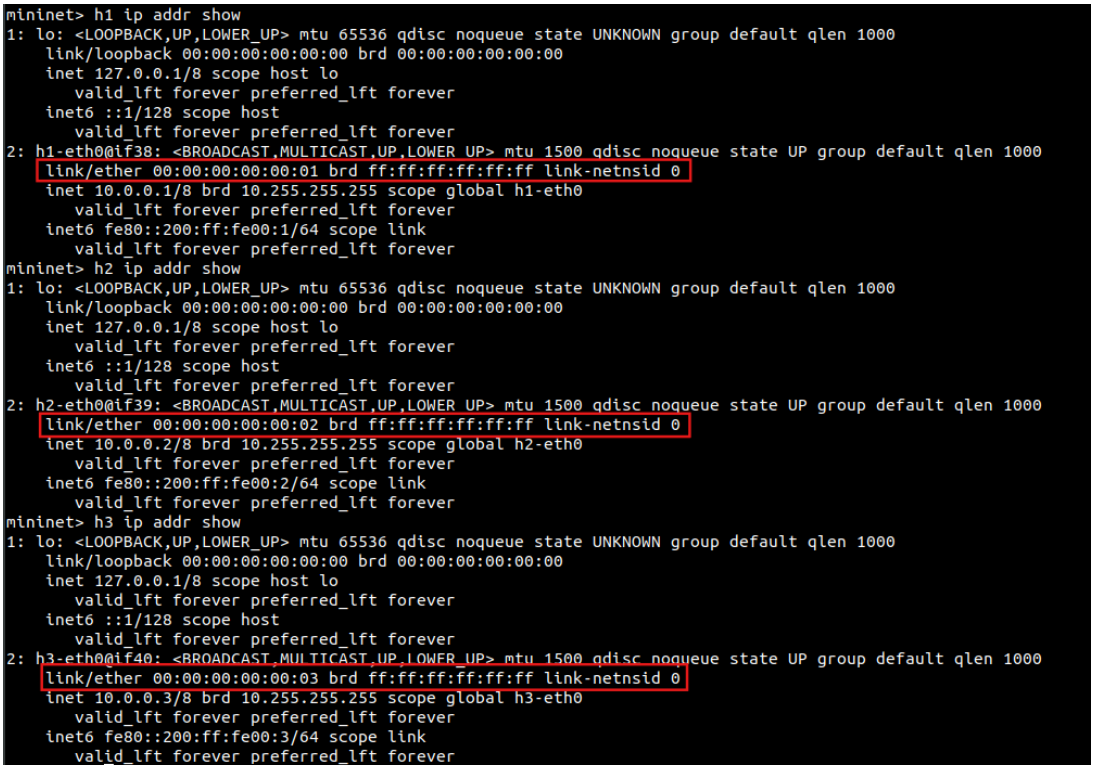
```
mininet> h1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1-eth0@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 5a:a2:1b:f3:16:2b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::58a2:1bff:fef3:162b/64 scope link
       valid_lft forever preferred_lft forever
mininet> h2 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h2-eth0@if28: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 72:f9:12:a9:d7:7b brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/8 brd 10.255.255.255 scope global h2-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::70f9:12ff:fea9:d77b/64 scope link
       valid_lft forever preferred_lft forever
mininet> h3 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h3-eth0@if29: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether b2:69:75:25:8a:a0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.3/8 brd 10.255.255.255 scope global h3-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::b069:75ff:fe25:8aa0/64 scope link
       valid_lft forever preferred_lft forever
```

Podemos observar que para *h1*, su dirección MAC es: 5a:a2:1b:f3:16:2b
La dirección MAC de *h2* es: 72:f9:12:a9:d7:7b
La dirección MAC de *h3* es: b2:69:75:25:8a:a0

Ahora, iniciamos la misma topología poniendo *--mac* al final:

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,3 --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

El número de switches y hosts es el mismo y están conectados de la misma manera.
Vemos las direcciones MAC de los tres hosts:



Podemos observar que para *h1*, su dirección MAC es: 00:00:00:00:00:01
La dirección MAC de *h2* es: 00:00:00:00:00:02
La dirección MAC de *h3* es: 00:00:00:00:00:03

¿CUÁLES SON LAS DIFERENCIAS?

Cuando no añadimos *--mac* al iniciar la topología, vemos que las direcciones MAC son:

h1 → 5a:a2:1b:f3:16:2b
h2 → 72:f9:12:a9:d7:7b
h3→ b2:69:75:25:8a:a0

Mientras que cuando sí lo añadimos:

h1 → 00:00:00:00:00:01
h2 → 00:00:00:00:00:02
h3 → 00:00:00:00:00:03

La diferencia es que cuando iniciamos la topología sin incluir *--mac*, la asignación de estas direcciones se hace de manera desordenada y aleatoria. Sin embargo, cuando añadimos *--mac* al final, la asignación es ordenada.

1.2.3:

Probamos a iniciar la topología con el switch user, pero nos da error:

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,3 --switch user
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
Cannot find required executable ofdatapath.
Please make sure that the OpenFlow reference user switch(openflow.org) is installed and available in your $PATH:
(/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin)
```

Sin embargo, con *OpenVSwitch* sí podemos hacerlo:

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,3 --switch ovsk
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

1.2.4:

Ahora vamos a lanzar mininet con una topología, pero esta vez especificando la versión de OpenFlow que queremos que se utilice en el *control plane*:
Vamos a lanzar mininet con la versión de OpenFlow 1.3:

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,1,4 --controller ovsc --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1s1 h2s1 h3s1 h4s1
*** Adding switches:
s1
*** Adding links:
(h1s1, s1) (h2s1, s1) (h3s1, s1) (h4s1, s1)
*** Configuring hosts
h1s1 h2s1 h3s1 h4s1
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> h2s1 h3s1 h4s1
h2s1 -> h1s1 h3s1 h4s1
h3s1 -> h1s1 h2s1 h4s1
h4s1 -> h1s1 h2s1 h3s1
*** Results: 0% dropped (12/12 received)
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
2025-03-26T17:27:39Z|00001|vconn|WARN|unix:/var/run/openvswitch/s1.mgmt: version negotiation failed (we support version 0x01, peer supports
version 0x04)
ovs-ofctl: s1: failed to connect to socket (Broken pipe)
```

El switch *s1* está configurado con OpenFlow 1.3, pero *dpctl*, sin especificarle una versión, trata de usar OpenFlow 1.0, por eso hay un fallo.

```
mininet> dpctl dump-flows -O OpenFlow13
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=55.207s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=c2:d9:3c:2e:86:e1,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=55.197s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=55.189s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.4,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=55.173s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=c2:d9:3c:2e
:86:e1,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=55.166s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=c2:d9:3c:2e
:86:e1,arp_spa=10.0.0.4,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=55.144s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=be:9f:8b:2d
:16:8c,arp_spa=10.0.0.4,arp_tpa=10.0.0.3,arp_op=2 actions=output:"s1-eth3"
 cookie=0x0, duration=50.180s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.4,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=50.176s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=50.174s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=c2:d9:3c:2e:86:e1,dl_dst=56:b0:eb:6b
:83:92,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=50.174s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=56:b0:eb:6b:83:92,dl_dst=6e:1f:8c:9d
:a7:65,arp_spa=10.0.0.1,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
 cookie=0x0, duration=50.174s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=56:b0:eb:6b:83:92,dl_dst=be:9f:8b:2d
:16:8c,arp_spa=10.0.0.1,arp_tpa=10.0.0.3,arp_op=2 actions=output:"s1-eth3"
 cookie=0x0, duration=50.173s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=56:b0:eb:6b:83:92,dl_dst=c2:d9:3c:2e
:86:e1,arp_spa=10.0.0.1,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=49.925s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=be:9f:8b:2d
:16:8c,arp_spa=10.0.0.4,arp_tpa=10.0.0.3,arp_op=1 actions=output:"s1-eth3"
 cookie=0x0, duration=49.925s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=c2:d9:3c:2e
:86:e1,arp_spa=10.0.0.4,arp_tpa=10.0.0.2,arp_op=1 actions=output:"s1-eth2"
 cookie=0x0, duration=49.925s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=c2:d9:3c:2e
:86:e1,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=1 actions=output:"s1-eth2"
 cookie=0x0, duration=49.924s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=6e:1f:8c:9d
:a7:65,arp_spa=10.0.0.3,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
 cookie=0x0, duration=49.924s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=c2:d9:3c:2e:86:e1,dl_dst=6e:1f:8c:9d
:a7:65,arp_spa=10.0.0.2,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
```

[...]

```
b:83:92,nw_src=10.0.0.4,nw_dst=10.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth1"
 cookie=0x0, duration=55.138s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=56:b0:eb:6b:83:92,dl_dst=6e:1f:8c:9
d:a7:65,nw_src=10.0.0.1,nw_dst=10.0.0.4,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth4"
 cookie=0x0, duration=55.131s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=c2:d9:3c:2
e:86:e1,nw_src=10.0.0.4,nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth2"
 cookie=0x0, duration=55.131s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=c2:d9:3c:2e:86:e1,dl_dst=6e:1f:8c:9
d:a7:65,nw_src=10.0.0.2,nw_dst=10.0.0.4,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth4"
 cookie=0x0, duration=55.127s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=6e:1f:8c:9d:a7:65,dl_dst=be:9f:8b:2
d:16:8c,nw_src=10.0.0.4,nw_dst=10.0.0.3,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth3"
 cookie=0x0, duration=55.127s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,icmp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=be:9f:8b:2d:16:8c,dl_dst=6e:1f:8c:9
d:a7:65,nw_src=10.0.0.3,nw_dst=10.0.0.4,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth4"
 cookie=0x0, duration=57.596s, table=0, n_packets=78, n_bytes=5708, priority=0 actions=CONTROLLER:128
mininet>
```

Ahora hemos forzado a *dpctl* a usar la versión 1.3 de OpenFlow y por eso sí ha funcionado.

Ahora repetimos el proceso sin el parámetro *protocols=OpenFlow13*

Con esto, no estamos forzando una versión específica de OpenFlow. Lo esperado sería que con *dpctl* sí funcione.

```
student@rs-2025:/usr/bin$ sudo mn --topo linear,1,4 --controller ovsc --switch ovs
*** Creating network
*** Adding controller
*** Adding hosts:
h1s1 h2s1 h3s1 h4s1
*** Adding switches:
s1
*** Adding links:
(h1s1, s1) (h2s1, s1) (h3s1, s1) (h4s1, s1)
*** Configuring hosts
h1s1 h2s1 h3s1 h4s1
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

```
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> h2s1 h3s1 h4s1
h2s1 -> h1s1 h3s1 h4s1
h3s1 -> h1s1 h2s1 h4s1
h4s1 -> h1s1 h2s1 h3s1
*** Results: 0% dropped (12/12 received)
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=14.525s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=4a:dd:1d:bf:e1
:19,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=14.517s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=3a:7c:ff:7a:c6
:f9,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=14.511s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e
:f8,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.4,arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=14.502s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=3a:7c:ff:7a:c6
:f9,dl_dst=4a:dd:1d:bf:e1:19,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=14.490s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e
:f8,dl_dst=4a:dd:1d:bf:e1:19,arp_spa=10.0.0.4,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=14.471s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e
:f8,dl_dst=3a:7c:ff:7a:c6:f9,arp_spa=10.0.0.4,arp_tpa=10.0.0.3,arp_op=2 actions=output:"s1-eth3"
 cookie=0x0, duration=9.458s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e:
f8,dl_dst=4a:dd:1d:bf:e1:19,arp_spa=10.0.0.4,arp_tpa=10.0.0.2,arp_op=1 actions=output:"s1-eth2"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e:
f8,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.4,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=3a:7c:ff:7a:c6:
f9,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.3,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=4a:dd:1d:bf:e1:
19,dl_dst=96:ab:64:35:df:b5,arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",vlan_tci=0x0000/0x1fff,dl_src=5e:d0:c5:22:6e:
f8,dl_dst=3a:7c:ff:7a:c6:f9,arp_spa=10.0.0.4,arp_tpa=10.0.0.3,arp_op=1 actions=output:"s1-eth3"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=3a:7c:ff:7a:c6:
f9,dl_dst=4a:dd:1d:bf:e1:19,arp_spa=10.0.0.3,arp_tpa=10.0.0.2,arp_op=1 actions=output:"s1-eth2"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=4a:dd:1d:bf:e1:
19,dl_dst=5e:d0:c5:22:6e:f8,arp_spa=10.0.0.2,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
 cookie=0x0, duration=9.454s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=96:ab:64:35:df:
b5,dl_dst=5e:d0:c5:22:6e:f8,arp_spa=10.0.0.1,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
 cookie=0x0, duration=9.453s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=96:ab:64:35:df:
b5,dl_dst=3a:7c:ff:7a:c6:f9,arp_spa=10.0.0.1,arp_tpa=10.0.0.3,arp_op=2 actions=output:"s1-eth3"
 cookie=0x0, duration=9.453s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=96:ab:64:35:df:
b5,dl_dst=4a:dd:1d:bf:e1:19,arp_spa=10.0.0.1,arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=9.453s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth3",vlan_tci=0x0000/0x1fff,dl_src=3
f9,dl_dst=5e:d0:c5:22:6e:f8,arp_spa=10.0.0.3,arp_tpa=10.0.0.4,arp_op=2 actions=output:"s1-eth4"
```



```
mininet> dpctl dump-flows -O OpenFlow13
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=295.597s, table=0, n_packets=92, n_bytes=6744, priority=0 actions=CONTROLLER:128
```

Podemos ver que con *dpctl* sin especificar versión, si funciona. Esto es así porque hemos quitado el parámetro *protocols=OpenFlow13.*
Con *dpctl* forzando la versión 1.3, solo nos muestra la regla por defecto que saldrá siempre.

¿CUÁLES ES LA DIFERENCIA EN LA SALIDA DE AMBOS dpctl dump-flows?

Según si se ha creado la topología especificando la versión de OpenFlow o no, habrá que usar un comando dpctl u otro.
En caso de que sí se haya especificado una versión de OpenFlow (en nuestro caso la 1.3) tendremos que usar *dpctl dump-flows -O OpenFlow13* para que *dpctl* y el switch estén alineados en la versión de OpenFlow.
En caso de que no se haya especificado la versión de OpenFlow al crear la topología estará usando la versión 1.0 por defecto. Esa versión sí puede trabajar con *dpctl dump-flows* sin especificar versión (trabaja automáticamente con la 1.0).

1.2.5:

Instalamos *xterm*.

Ejecutamos el comando:

```
root@rs-2025:/usr/bin# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1s1-eth0@if25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue stat
e UP group default qlen 1000
    link/ether e2:40:cc:fb:8e:ba brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1s1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::e040:ccff:fefb:8eba/64 scope link
       valid_lft forever preferred_lft forever
root@rs-2025:/usr/bin#
```

La dirección ip es: 10.0.0.0.1/8

```
root@rs-2025:/usr/bin# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h2s1-eth0@if26: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 56:eb:70:f2:59:3c brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/8 brd 10.255.255.255 scope global h2s1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::54eb:70ff:fef2:593c/64 scope link
       valid_lft forever preferred_lft forever
root@rs-2025:/usr/bin#
```

La dirección ip es: 10.0.0.2/8

Desde el terminal de *h1* hacemos un ping a *h2*:

```
root@rs-2025:/usr/bin# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.49 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.286 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.132 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.075 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.100 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6127ms
rtt min/avg/max/mdev = 0.049/0.462/2.491/0.831 ms
root@rs-2025:/usr/bin#
```

Hito 2:

1.3 Programando Mininet en Python

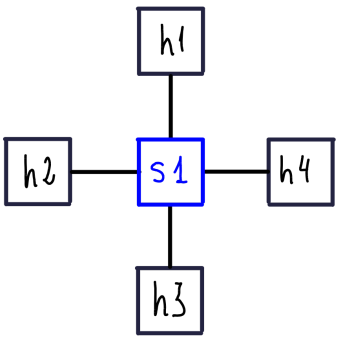
1.3.1:

```
student@rs-2025:~/Desktop/Devel$ cd /home/student/Desktop/Devel
student@rs-2025:~/Desktop/Devel$ chmod +x mininet1.py
student@rs-2025:~/Desktop/Devel$ sudo ./mininet1.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
student@rs-2025:~/Desktop/Devel$
```

Vemos que la topología tiene:

- 4 hosts
- 1 switch
- 1 controller

La topología tiene forma de cruz (estrella):



Hito 3:

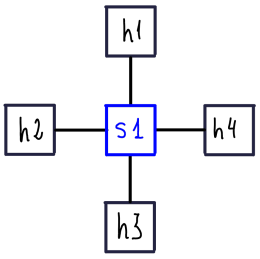
1.3.2:

```
student@rs-2025:~$ cd /home/student/Desktop/Devel
student@rs-2025:~/Desktop/Devel$ chmod +x mininet2.py
student@rs-2025:~/Desktop/Devel$ sudo ./mininet2.py
[sudo] password for student:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Podemos ver que la topología consiste de:

- 4 hosts
- 1 switch
- 1 controlador

Están conectados en cruz (estrella):



Tras hacer el *pingall* podemos ver que todos los equipos están bien conectados entre sí.

Probamos a ejecutarlo con el parámetro *--mac*:

```
student@rs-2025:~/Desktop/Devel$ sudo ./mininet2.py --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1-eth0@if49: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 32:2f:30:45:f7:16 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::302f:30ff:fe45:f716/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

Podemos ver que no nos ha permitido controlar la dirección MAC.

1.3.3:

```python
#!/usr/bin/env python3


from mininet.net import Mininet
from mininet.topo import Topo
from mininet.log import setLogLevel
from mininet.cli import CLI


class SingleSwitchTopo(Topo):
    'My single switch connected to n hosts.'

    def build(self, N=2):
        switch = self.addSwitch('s1')
        for hn in range(N):
            host = self.addHost(f'h{hn+1}', mac=f'00:00:00:00:00:0{hn+1}')
            self.addLink(host, switch)


def simpleTestCLI():
    topo = SingleSwitchTopo(4)
    net = Mininet(topo)
    net.start()
    CLI(net)
    net.stop()


if __name__ == '__main__':
    # Tell mininet to print useful information
    setLogLevel('info')
    simpleTestCLI()
```

Añadimos el parámetro *mac* y le asignamos un valor variable que siga la estructura de una dirección MAC. Tal y como está hecho el código, solo funciona en topologías de hasta 9 hosts.

```
student@rs-2025:~/Desktop/Devel$ cd /home/student/Desktop/Devel
student@rs-2025:~/Desktop/Devel$ sudo ./mininet3.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Ahora vemos cada uno de los hosts:

```
student@rs-2025:~/Desktop/Devel$ cd /home/student/Desktop/Devel
student@rs-2025:~/Desktop/Devel$ sudo ./mininet3.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h1-eth0@if73: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:00:00:01 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe00:1/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

```
mininet> h2 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h2-eth0@if74: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:00:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.2/8 brd 10.255.255.255 scope global h2-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe00:2/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

```
mininet> h3 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h3-eth0@if75: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:00:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.3/8 brd 10.255.255.255 scope global h3-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe00:3/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

```
mininet> h4 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: h4-eth0@if76: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:00:00:00:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.0.4/8 brd 10.255.255.255 scope global h4-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::200:ff:fe00:4/64 scope link
       valid_lft forever preferred_lft forever
mininet>
```

Hito 4:

2.2 Interacción básica entre Ryu y la red:

El fichero *mininet4.py* nos queda así:

```python
#!/usr/bin/env python3


from mininet.net import Mininet
from mininet.topo import Topo
from mininet.log import setLogLevel
from mininet.cli import CLI
from mininet.node import RemoteController, OVSSwitch
from functools import partial


class SingleSwitchTopo(Topo):
    'My single switch connected to n hosts.'

    def build(self, N=2):
        switch = self.addSwitch('s1')
        for hn in range(N):
            host = self.addHost(f'h{hn+1}', mac=f'00:00:00:00:00:0{hn+1}')
            self.addLink(host, switch)


def simpleTestCLI():
    topo = SingleSwitchTopo(4)
    net = Mininet(topo,
        controller=partial(RemoteController, ip="127.0.0.1"),
        switch=partial(OVSSwitch, protocols="OpenFlow13"))
    net.start()
    CLI(net)
    net.stop()


if __name__ == '__main__':
    # Tell mininet to print useful information
    setLogLevel('info')
    simpleTestCLI()
```

Ahora lanzamos el controlador Ryu con la clase *ryu.app.simple_switch_13*. Lanzamos también *mininet4.py* y vemos lo siguiente:

```
student@rs-2025:~$ ryu-manager ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 0000000000000001 00:00:00:00:00:03 33:33:ff:00:00:03 3
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 0000000000000001 00:00:00:00:00:02 33:33:ff:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:01 33:33:ff:00:00:01 1
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:16 4
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:02 4
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:16 2
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:16 4
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:16 3
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:16 1
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:02 4
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:02 4
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:02 4
packet in 0000000000000001 00:00:00:00:00:01 33:33:00:00:00:02 1
packet in 0000000000000001 00:00:00:00:00:02 33:33:00:00:00:02 2
packet in 0000000000000001 00:00:00:00:00:04 33:33:00:00:00:02 4
packet in 0000000000000001 00:00:00:00:00:03 33:33:00:00:00:02 3
packet in 0000000000000001 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 1
packet in 0000000000000001 00:00:00:00:00:02 00:00:00:00:00:01 2
packet in 0000000000000001 00:00:00:00:00:01 00:00:00:00:00:02 1
```

```
student@rs-2025:~/Devel$ sudo ./mininet4.py
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

Podemos ver que sí se ha establecido la conexión. Y hemos visto que aumentaban los registros al hacer el *pingall.*

Ahora lanzamos *ryu.app.simple_switch_12* junto con *mininet4.py*:

```
student@rs-2025:~$ ryu-manager ryu.app.simple_switch_12
loading app ryu.app.simple_switch_12
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_12 of SimpleSwitch12
instantiating app ryu.controller.ofp_handler of OFPHandler
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44048)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44054)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44056)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44068)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44070)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44086)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44102)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 44106)
unsupported version 0x4. If possible, set the switch to use one of the versions [3] on datapath ('127.0.0.1', 53090)
```

Vemos que no se establece la conexión ya que cada uno espera una versión de OpenFlow distinta.

2.3 Programas de Ryu personalizados:

No hemos sido capaces de ejecutar el código.