

# Capítulo 1

## Análisis script meshAP.py

### 1.1. Configuración del script meshAP.py

Este fichero contiene la configuración de una red mesh simple que cuenta con dos access points y dos estaciones base. Dicha configuración corresponde a la de una red SDN en la que los access points funcionan como switches típicos de este tipo de redes.

El contenido del script meshAP.py es el siguiente:

```
1 #!/usr/bin/python
2
3 '''
4 This example shows on how to create wireless link between two APs
5 with mesh
6 The wireless mesh network is based on IEEE 802.11s
7 '''
8
9 from mininet.log import setLogLevel, info
10 from mn_wifi.link import wmediumd, mesh
11 from mn_wifi.cli import CLI
12 from mn_wifi.net import Mininet_wifi
13 from mn_wifi.wmediumdConnector import interference
14
15
16 def topology():
17     'Create a network.'
18     net = Mininet_wifi(link=wmediumd, wmediumd_mode=interference)
19
```

```

20 info('*** Creating nodes\n')
21 sta1 = net.addStation('sta1', mac='00:00:00:00:00:11',
22     position='1,1,0')
23 sta2 = net.addStation('sta2', mac='00:00:00:00:00:12',
24     position='31,11,0')
25 ap1 = net.addAccessPoint('ap1', wlans=2, ssid='ssid1',
26     position='10,10,0')
27 ap2 = net.addAccessPoint('ap2', wlans=2, ssid='ssid2',
28     position='30,10,0')
29 c0 = net.addController('c0')
30
31 info('*** Configuring wifi nodes\n')
32 net.configureWifiNodes()
33
34 info('*** Associating Stations\n')
35 net.addLink(sta1, ap1)
36 net.addLink(sta2, ap2)
37 net.addLink(ap1, intf='ap1-wlan2', cls=mesh, ssid='mesh-ssid',
38     channel=5)
39 net.addLink(ap2, intf='ap2-wlan2', cls=mesh, ssid='mesh-ssid',
40     channel=5)
41
42 info('*** Starting network\n')
43 net.build()
44 c0.start()
45 ap1.start([c0])
46 ap2.start([c0])
47
48 info('*** Running CLI\n')
49 CLI(net)
50
51 info('*** Stopping network\n')
52 net.stop()
53
54
55 if __name__ == '__main__':
56     setLogLevel('info')
57     topology()

```

Se trata de una red mesh de medio inalámbrico (*link = wmedium*) con interferencias (*wmedium\_mode = interference*). El simulador se encarga de calcular el nivel de interferencia en base a la distancia existente entre un nodo y sus nodos adyacentes. También se desprende del

script que la red presenta el controlador propio de las redes definidas por software, conocidas como redes SDN por sus siglas en inglés.

Los dos puntos de acceso (*access points* (APs) en adelante) de la red mesh configurada en el script presentan cada uno de ellos dos WLANs, una de ellas con configuración mesh. Además se indica las direcciones MAC de las dos estaciones base *sta1* y *sta2*.

Respecto a los enlaces entre los diferentes nodos de la red, se observa que se configura un enlace entre *ap1* y *ap2* y otro entre *sta2* y *ap2*. En *ap1* y *ap2* además se configuran las interfaces '*ap1-wlan2*' y '*ap2-wlan2*' como enlaces mesh en el canal 5 de la red.

## 1.2. Topología inicial

En la Figura 1.1 se representa la posición de los nodos y APs de la red, obtenida mediante el módulo de representación que ofrece el entorno Mininet Wifi.

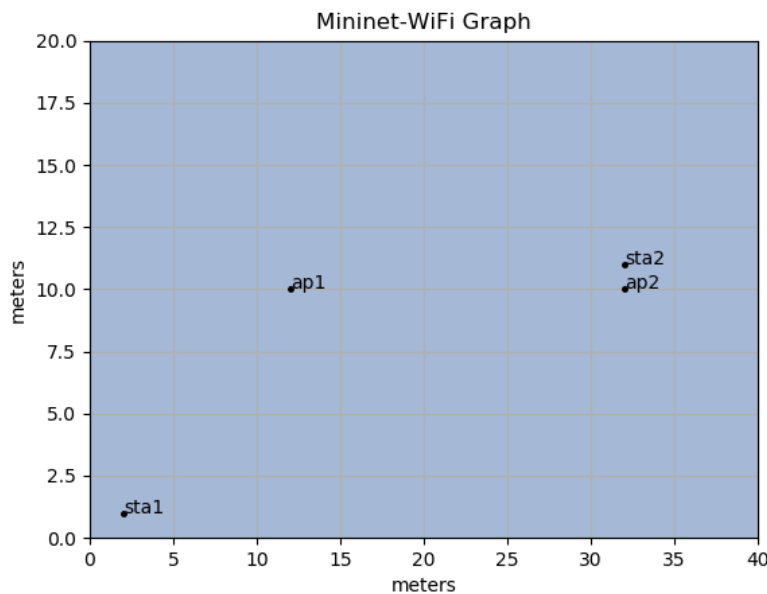


Figura 1.1: Posición de los nodos de la red meshAP.

Con la información que se tiene hasta el momento (la que aporta el script *meshAP.py* y la posición de los equipos de la red), se puede determinar que la topología de la red mesh es la indicada en la Figura 1.2, a falta de conocer cuáles son las interfaces que se conectan entre los diferentes equipos.

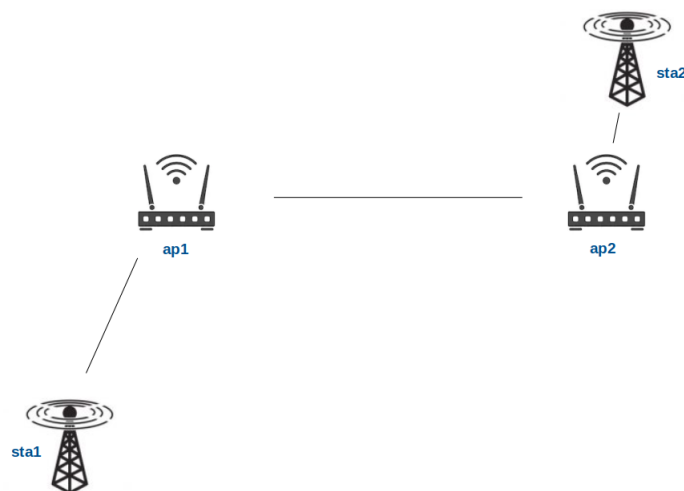


Figura 1.2: Topología de la red meshAP.

Sin embargo, aún falta conocer cuáles son las interfaces de los nodos, cómo se conectan entre ellas y qué papel tiene en estas conexiones el controlador de la red. En los siguientes apartados del presente capítulo se analizará el funcionamiento de la red en ejecución para comprender cómo funciona realmente una red mesh definida por software y poder completar el esquema de la topología de la red analizada.

### 1.3. Establecimiento de la conexión y configuración de las interfaces

### 1.4. Análisis de los flujos de datos con `pingall`

Hay que recordar que el comando `pingall` se encarga de ejecutar un mensaje *ping* entre todas las estaciones base que componen una red. En la red meshAP, por tanto, se generarán 2 *pings*: uno desde `sta1` hacia `sta2`, y otro en el sentido contrario.

Los mensajes *ping* funcionan de la siguiente manera: la estación origen envía un paquete ICMP `echo request` hacia la estación destino, que debe contestar con otro mensaje ICMP `echo reply`. Si la estación base origen no conoce la dirección MAC de la estación destino,

además se generan mensajes ARP request y ARP reply. En el ARP request, la estación origen envía el mensaje a todos los nodos (estaciones base, puntos de acceso, *hosts*) a los que se encuentra conectada, es decir, lo envía a la dirección de broadcast (ff:ff:ff:ff:ff:ff). En este mensaje, origen pregunta quién tiene la dirección IP destino del ICMP echo request, y el nodo al que pertenezca dicha IP responde con su dirección MAC en un mensaje ARP reply a la estación origen. Una vez conocida la dirección MAC del destino, el origen envía el ICMP echo request. Cuando el destino lo recibe, puede ocurrir que tampoco tenga la dirección MAC del origen, si se ha borrado de su caché, por lo que tendría que realizar el mismo procedimiento con los mensajes ARP para conocerla y, finalmente, enviar su respuesta ICMP echo reply a esa dirección.

Sin embargo, en las redes SDN el proceso de mensajes se ve un poco alterado. Cuando sta1 envía su ARP request hacia la dirección de broadcast para conocer la dirección MAC de sta2, en primer lugar el mensaje llega al *access point* ap1. Seguidamente, ap1 deberá consultar su tabla de flujos para comprobar si existe alguna entrada para tráfico ARP request procedente de sta1 y destinado a sta2. Si tiene alguna entrada que coincida para este tipo de tráfico, sigue las normas o acciones que se indiquen en ella. Si por el contrario, no la tiene, deberá consultar al controlador de la red qué debe hacer en ese caso.

Entre los *access points* y el controlador de una red SDN se intercambian distintos tipos de mensajes. Normalmente el *access point* consulta qué hacer y el controlador le indica a quién debe enviar el tráfico, por qué puerto, si debe eliminarlo, o si tiene que añadir una nueva entrada para ese tipo de tráfico en su tabla de flujos. Estos mensajes entre el controlador y los *access points* se analizan con más detalle en el Apartado 1.5 del presente capítulo.

A continuación se analiza el contenido de las tablas de flujos del controlador c0 de la red, y de los puntos de acceso ap1 y ap2. Como se ha indicado, estas tablas deberían tener entradas para todos los tipos de tráfico que se envían entre los nodos, tras ejecutar el comando pingall.

El datapath del controlador de la red tiene los siguientes puertos e interfaces:

```
system@ovs-system:
```

```
lookups: hit:79 missed:37 lost:0
flows: 0
masks: hit:153 total:0 hit/pkt:1.32
port 0: ovs-system (internal)
port 1: ap1-wlan1
```

```
port 2: ap1-wlan2
port 3: ap1-mp2
port 4: ap1 (internal)
port 5: ap2-wlan1
port 6: ap2-wlan2
port 7: ap2-mp2
port 8: ap2 (internal)
```

Para comprobar los flujos que se instalan en el controlador, se debe ejecutar el siguiente comando:

```
mininet-wifi>sh ovs-dpctl dump-flows
```

```
recirc_id(0),in_port(5),eth(src=00:00:00:00:00:12,dst=00:00:00:00:00:11),
eth_type(0x0800),ipv4(src=10.0.0.2,dst=10.0.0.1,proto=1,tos=0/0xfc,frag=no),
icmp(type=8,code=0), packets:0, bytes:0, used:never, actions:7
```

Regla 1: indica que todo el tráfico ICMP ECHO (icmp\_type=8, icmp\_code=0) que entre por el puerto 5 del controlador (interfaz ap2-wlan1), con direcciones MAC origen y destino 00:00:00:00:00:12 y 00:00:00:00:00:11, respectivamente, y direcciones IP 10.0.0.2 y 10.0.0.1, origen y destino, salga por el puerto 7, interfaz ap2-mp2.

```
recirc_id(0),in_port(7),eth(src=00:00:00:00:00:11,dst=00:00:00:00:00:12),
eth_type(0x0800),ipv4(src=10.0.0.1,dst=10.0.0.2,proto=1,tos=0/0xfc,frag=no),
icmp(type=0,code=0), packets:0, bytes:0, used:never, actions:5
```

Regla 2: indica que todo el tráfico ICMP echo reply (icmp\_type=0, icmp\_code=0) que entre por el puerto 7 del controlador (interfaz ap2-mp2), con direcciones MAC origen y destino 00:00:00:00:00:11 y 00:00:00:00:00:12, respectivamente, y direcciones IP 10.0.0.1 y 10.0.0.2, origen y destino, salga por el puerto 5, interfaz ap2-wlan1.

```
recirc_id(0),in_port(3),eth(src=00:00:00:00:00:12,dst=00:00:00:00:00:11),
eth_type(0x0800),ipv4(src=10.0.0.2,dst=10.0.0.1,proto=1,tos=0/0xfc,frag=no),
icmp(type=8,code=0), packets:0, bytes:0, used:never, actions:1
```

Regla 3: indica que todo el tráfico ICMP ECHO (icmp\_type=8, icmp\_code=0) que entre por el puerto 3 del controlador (interfaz ap1-mp2), con direcciones MAC origen y destino 00:00:00:00:00:12 y 00:00:00:00:00:11, respectivamente, y direcciones IP 10.0.0.2 y 10.0.0.1, origen y destino, salga por el puerto 1, interfaz ap1-wlan1.

```
recirc_id(0),in_port(1),eth(src=00:00:00:00:00:11,dst=00:00:00:00:00:12),  
eth_type(0x0800),ipv4(src=10.0.0.1,dst=10.0.0.2,proto=1,tos=0/0xfc,frag=  
no), icmp(type=0,code=0), packets:0, bytes:0, used:never, actions:3
```

Regla 4: indica que todo el tráfico ICMP echo reply (icmp\_type=0, icmp\_code=0) que entre por el puerto 1 del controlador (interfaz ap1-wlan1), con direcciones MAC origen y destino 00:00:00:00:00:11 y 00:00:00:00:00:12, respectivamente, y direcciones IP 10.0.0.1 y 10.0.0.2, origen y destino, salga por el puerto 3, interfaz ap1-mp2.

```
recirc_id(0),in_port(3),eth(src=00:00:00:00:00:12,dst=00:00:00:00:00:11),  
eth_type(0x0806),arp(sip=10.0.0.2,tip=10.0.0.1, op=1/0xff), packets:0,  
bytes:0, used:never, actions:1
```

Regla 5: indica que todo el tráfico ARP request (op=1) que entre por el puerto 3 del controlador (interfaz ap1-mp2), con direcciones MAC origen y destino 00:00:00:00:00:12 y 00:00:00:00:00:11, respectivamente, y direcciones IP 10.0.0.2 y 10.0.0.1, origen y destino, salga por el puerto 1, interfaz ap1-wlan1.

```
recirc_id(0),in_port(5),eth(src=00:00:00:00:00:12,dst=00:00:00:00:00:11),  
eth_type(0x0806),arp(sip=10.0.0.2,tip=10.0.0.1,op=1/0xff), packets:0,  
bytes:0, used:never, actions:7
```

Regla 6: indica que todo el tráfico ARP request (op=1) que entre por el puerto 5 del controlador (interfaz ap2-wlan1), con direcciones MAC origen y destino 00:00:00:00:00:12 y 00:00:00:00:00:11, respectivamente, y direcciones IP 10.0.0.2 y 10.0.0.1, origen y destino, salga por el puerto 7, interfaz ap2-mp2.

Para ver la tabla de flujos del *access point* ap1, hay que ejecutar el siguiente comando:

```
mininet-wifi>sh ovs-ofctl dump-flows ap1
```

```
cookie=0x0, duration=5.054s, table=0, n_packets=0, n_bytes=0, idle_timeout  
=60, priority=65535, arp, in_port='ap1-mp2',vlan_tci=0x0000,dl_src=00:00:00:  
00:00:12,dl_dst=00:00:00:00:00:11, arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_  
op=2 actions=output:'ap1-wlan1'
```

Los paquetes ARP reply (arp\_op=2) que entran a ap1 por su interfaz 'ap1-mp2', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de

ARP 10.0.0.2 y destino ARP 10.0.0.1, saldrán por la interfaz ‘ap1-wlan1’ de ap1. No corresponde con ninguna regla del controlador, probablemente ap1 aprenda de la regla instalada para el ARP request.

```
cookie=0x0, duration=9.892s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, arp, in_port='ap1-mp2', vlan_tci=0x0000, dl_src=00:00:00:00:00:12, dl_dst=00:00:00:00:00:11, arp_spa=10.0.0.2, arp_tpa=10.0.0.1, arp_op=1 actions=output:'ap1-wlan1'
```

Los paquetes ARP request (arp\_op=1) que entran a ap1 por su interfaz ‘ap1-mp2’, cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ARP 10.0.0.2 y destino ARP 10.0.0.1, saldrán por la interfaz ‘ap1-wlan1’ de ap1. Corresponde con la regla 5 del controlador.

```
cookie=0x0, duration=9.889s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, arp, in_port='ap1-wlan1', vlan_tci=0x0000, dl_src=00:00:00:00:00:11, dl_dst=00:00:00:00:00:12, arp_spa=10.0.0.1, arp_tpa=10.0.0.2, arp_op=2 actions=output:'ap1-mp2'
```

Los paquetes ARP reply (arp\_op=2) que entran a ap1 por su interfaz ‘ap1-wlan1’, cuyo origen es la dirección MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ARP 10.0.0.1 y destino ARP 10.0.0.2, saldrán por la interfaz ‘ap1-mp2’ de ap1. No corresponde con ninguna regla del controlador, probablemente ap1 aprenda de la regla instalada para el ARP request.

```
cookie=0x0, duration=5.051s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp, in_port='ap1-wlan1', vlan_tci=0x0000, dl_src=00:00:00:00:00:11, dl_dst=00:00:00:00:00:12, nw_src=10.0.0.1, nw_dst=10.0.0.2, nw_tos=0, icmp_type=8, icmp_code=0 actions=output:'ap1-mp2'
```

Los paquetes ICMP ECHO (icmp\_type=8, icmp\_code=0) que entran a ap1 por su interfaz ‘ap1-wlan1’, cuyo origen es la dirección MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ICMP 10.0.0.1 y destino ICMP 10.0.0.2, saldrán por la interfaz ‘ap1-mp2’ de ap1.

```
cookie=0x0, duration=5.036s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp, in_port='ap1-mp2', vlan_tci=0x0000, dl_src=00:00:00:00:00:12, dl_dst=00:00:00:00:00:11, nw_src=10.0.0.2, nw_dst=10.0.0.1, nw_tos=0, icmp_type=0, icmp_code=0 actions=output:'ap1-wlan1'
```

Los paquetes ICMP echo reply (icmp\_type=0, icmp\_code=0) que entran a ap1 por su inter-



faz 'ap1-mp2', con dirección MAC origen 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ICMP 10.0.0.2 y destino ICMP 10.0.0.1, saldrán por la interfaz 'ap1-wlan1' de ap1.

```
cookie=0x0, duration=5.027s, table=0, n_packets=0, n_bytes=0, idle_timeout=60,
priority=65535, icmp, in_port='ap1-mp2', vlan_tci=0x0000, dl_src=00:00:00:
:00:00:12, dl_dst=00:00:00:00:00:11, nw_src=10.0.0.2, nw_dst=10.0.0.1, nw_tos=0,
icmp_type=8, icmp_code=0 actions=output:'ap1-wlan1'
```

Los paquetes ICMP ECHO (icmp\_type=8, icmp\_code=0) que entran a ap1 por su interfaz 'ap1-mp2', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ICMP 10.0.0.2 y destino ICMP 10.0.0.1, saldrán por la interfaz 'ap1-wlan1' de ap1.

```
cookie=0x0, duration=5.024s, table=0, n_packets=0, n_bytes=0, idle_timeout
=60, priority=65535, icmp, in_port='ap1-wlan1', vlan_tci=0x0000, dl_src=00:00:00:
00:00:11, dl_dst=00:00:00:00:00:12, nw_src=10.0.0.1, nw_dst=10.0.0.2, nw_tos=0,
icmp_type=0, icmp_code=0 actions=output:'ap1-mp2'
```

Los paquetes ICMP echo reply (icmp\_type=0, icmp\_code=0) que entran a ap1 por su interfaz 'ap1-wlan1', CON origen MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ICMP 10.0.0.1 y destino ICMP 10.0.0.2, saldrán por la interfaz 'ap1-mp2' de ap1.

El mismo comando se ejecuta para ap2:

```
mininet-wifi>sh ovs-ofctl dump-flows ap2
```

Los flujos que se obtienen para ap2 son los siguientes:

```
cookie=0x0, duration=6.802s, table=0, n_packets=0, n_bytes=0, idle_timeout
=60, priority=65535, arp, in_port='ap2-wlan1', vlan_tci=0x0000, dl_src=00:00:00:
00:00:12, dl_dst=00:00:00:00:00:11, arp_spa=10.0.0.2, arp_tpa=10.0.0.1, arp_op=2
actions=output:'ap2-mp2'
```

Los paquetes ARP reply (arp\_op=2) que entran a ap2 por su interfaz 'ap2-wlan1', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ARP 10.0.0.2 y destino ARP 10.0.0.1, saldrán por la interfaz 'ap2-mp2' de ap2.

```
cookie=0x0, duration=1.531s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, arp,in_port='ap2-wlan1',vlan_tci=0x0000,dl_src=00:00:00:00:00:12,dl_dst=00:00:00:00:00:11, arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=1 actions=output:'ap2-mp2'
```

Los paquetes ARP request (arp\_op=1) que entran a ap2 por su interfaz 'ap2-wlan1', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ARP 10.0.0.2 y destino ARP 10.0.0.1, saldrán por la interfaz 'ap2-mp2' de ap2.

```
cookie=0x0, duration=1.520s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535,arp, in_port='ap2-mp2',vlan_tci=0x0000,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:12, arp_spa=10.0.0.1,arp_tpa=10.0.0.2,arp_op=2 actions=output:'ap2-wlan1'
```

Los paquetes ARP reply (arp\_op=2) que entran a ap2 por su interfaz 'ap2-mp2', cuyo origen es la dirección MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ARP 10.0.0.1 y destino ARP 10.0.0.2, saldrán por la interfaz 'ap2-wlan1' de ap2.

```
cookie=0x0, duration=6.788s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp,in_port='ap2-mp2',vlan_tci=0x0000,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:12, nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0, icmp_type=8,icmp_code=0 actions=output:'ap2-wlan1'
```

Los paquetes ICMP ECHO (icmp\_type=8, icmp\_code=0, nw\_tos=0) que entran a ap2 por su interfaz 'ap2-mp2', cuyo origen es la dirección MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ICMP 10.0.0.1 y destino ICMP 10.0.0.2, saldrán por la interfaz 'ap2-wlan1' de ap2.

```
cookie=0x0, duration=6.784s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp,in_port='ap2-wlan1',vlan_tci=0x0000,dl_src=00:00:00:00:00:12,dl_dst=00:00:00:00:00:11, nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0, icmp_type=0,icmp_code=0 actions=output:'ap2-mp2'
```

Los paquetes ICMP echo reply (icmp\_type=0, icmp\_code=0, nw\_tos=0) que entran a ap2 por su interfaz 'ap2-wlan1', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ICMP 10.0.0.2 y destino ICMP 10.0.0.1, saldrán por la interfaz 'ap2-mp2' de ap2.

```
cookie=0x0, duration=6.773s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp,in_port='ap2-wlan1',vlan_tci=0x0000,dl_src=00:00:00:00:00:12,dl_dst=00:00:00:00:00:11, nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,
```

```
icmp_type=8,icmp_code=0 actions=output:'ap2-mp2'
```

Los paquetes ICMP ECHO (icmp\_type=8, icmp\_code=0, nw\_tos=0) que entran a ap2 por su interfaz 'ap2-wlan1', cuyo origen es la dirección MAC 00:00:00:00:00:12 y destino 00:00:00:00:00:11, con dirección origen de ICMP 10.0.0.2 y destino ICMP 10.0.0.1, saldrán por la interfaz 'ap2-mp2' de ap2.

```
cookie=0x0, duration=6.763s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, icmp,in_port='ap2-mp2',vlan_tci=0x0000,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:12, nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0, icmp_type=0,icmp_code=0 actions=output:'ap2-wlan1'
```

Los paquetes ICMP echo reply (icmp\_type=0, icmp\_code=0, nw\_tos=0) que entran a ap2 por su interfaz 'ap2-mp2', cuyo origen es la dirección MAC 00:00:00:00:00:11 y destino 00:00:00:00:00:12, con dirección origen de ICMP 10.0.0.1 y destino ICMP 10.0.0.2, saldrán por la interfaz 'ap2-wlan1' de ap2.

## 1.5. Capturando el tráfico OpenFlow entre el controlador y los *access points*

Para analizar el intercambio de paquetes capturados cuando se ejecuta el comando pingall se necesita conocer los puertos configurados en los *access points* ap1 y ap2. El comando que nos indica el listado de los puertos de un *access point* es (aplicado para ap1):

```
sh ovs-ofctl show ap1
```

Tras ejecutarlo, en ap1 se obtiene la siguiente salida:

```
OFPT_FEATURES_REPLY (xid=0x2): dpid:10000000000000001 n_tables:254,
  n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
  ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src
  mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
1(ap1-wlan1): addr:02:00:00:00:02:00
  config: 0
  state: 0
```

```

    speed: 0 Mbps now, 0 Mbps max
2(ap1-wlan2): addr:02:00:00:00:03:00
    config: PORT_DOWN
    state: LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
3(ap1-mp2): addr:02:00:00:00:03:00
    config: 0
    state: 0
    speed: 0 Mbps now, 0 Mbps max
LOCAL(ap1): addr:7a:20:a9:fc:24:43
    config: PORT_DOWN
    state: LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0

```

En ap2:

```

OFPT_FEATURES_REPLY (xid=0x2): dpid:1000000000000002 n_tables:254,
    n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
    ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src
    mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
1(ap2-wlan1): addr:02:00:00:00:04:00
    config: 0
    state: 0
    speed: 0 Mbps now, 0 Mbps max
2(ap2-wlan2): addr:02:00:00:00:05:00
    config: PORT_DOWN
    state: LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
3(ap2-mp2): addr:02:00:00:00:05:00
    config: 0
    state: 0
    speed: 0 Mbps now, 0 Mbps max

```

```

LOCAL(ap2): addr:f6:95:f1:7c:34:4a
config: PORT_DOWN
state: LINK_DOWN
speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0

```

Tras capturar los paquetes en la interfaz loopback:lo, se realiza el análisis que sigue.

El primer ping que se realiza es desde sta1 hacia sta2, por lo que ap1 es el primer punto de acceso que se comunica con el controlador. En un paquete OFPT\_PACKET\_IN, ap1 consulta qué debe hacer con un paquete procedente la dirección MAC 00:00:00:00:00:11 dirigido a la dirección de broadcast, y que entra por el puerto 1 de ap1, correspondiente a la interfaz ap1-wlan1. En la Figura 1.3 se aprecia dicha interacción.

No.	Time	Source	Destination	Protocol	Length	Info
559	91.0...	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
560	91.0...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=
561	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	126	Type: OFPT_PACKET_IN
562	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	132	Type: OFPT_PACKET_OUT
563	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=
564	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	126	Type: OFPT_PACKET_IN
565	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	132	Type: OFPT_PACKET_OUT
566	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=
567	93.3...	00:00:00_00:00:12	00:00:00_00:00:...	OpenFlow	126	Type: OFPT_PACKET_IN
568	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
569	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=
570	93.3...	00:00:00_00:00:12	00:00:00_00:00:...	OpenFlow	132	Type: OFPT_PACKET_OUT
571	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=
572	93.3...	00:00:00_00:00:12	00:00:00_00:00:...	OpenFlow	126	Type: OFPT_PACKET_IN
573	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD

▶ Frame 561: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface  
 ▶ Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 48942, Dst Port: 6653, Seq: 6598, Ack: 6344,  
 ▼ OpenFlow 1.0  
   .000 0001 = Version: 1.0 (0x01)  
   Type: OFPT\_PACKET\_IN (10)  
   Length: 60  
   Transaction ID: 0  
   Buffer Id: 0xfffffffff  
   Total length: 42  
   In port: 1  
   Reason: No matching flow (table-miss flow entry) (0)  
   Pad: 00  
 ▶ Ethernet II, Src: 00:00:00\_00:00:11 (00:00:00:00:00:11), Dst: Broadcast (ff:ff:ff:f  
 ▼ Address Resolution Protocol (request)  
   Hardware type: Ethernet (1)  
   Protocol type: IPv4 (0x0800)  
   Hardware size: 6  
   Protocol size: 4  
   Opcode: request (1)  
   Sender MAC address: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
   Sender IP address: 10.0.0.1  
   Target MAC address: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
   Target IP address: 10.0.0.2

Figura 1.3: Consulta de ap1 al controlador

En su respuesta, el controlador indica mediante un paquete OFPT\_PACKET\_OUT que ap1 debe enviar ese tipo de tráfico por el puerto 65531, comportándose como un switch normal al que le llega un paquete dirigido a la dirección de broadcast. En la Figura 1.4 se muestra el paquete

capturado,

No.	Time	Source	Destination	Protocol	Length	Info
559	91.0...	127.0.0.1	127.0.0.1	OpenFlow	74	Type: OFPT_ECHO_REPLY
560	91.0...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6598 Ack=
✓ 561	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	126	Type: OFPT_PACKET_IN
562	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	132	Type: OFPT_PACKET_OUT
563	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6658 Ack=
564	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	126	Type: OFPT_PACKET_IN
565	93.3...	00:00:00_00:00:11	Broadcast	OpenFlow	132	Type: OFPT_PACKET_OUT
566	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6478 Ack=
567	93.3...	00:00:00_00:00:12	00:00:00_00:00:00...	OpenFlow	126	Type: OFPT_PACKET_IN
568	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
569	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6538 Ack=
570	93.3...	00:00:00_00:00:12	00:00:00_00:00:00...	OpenFlow	132	Type: OFPT_PACKET_OUT
571	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6538 Ack=
572	93.3...	00:00:00_00:00:12	00:00:00_00:00:00...	OpenFlow	126	Type: OFPT_PACKET_IN
573	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD

```

▶ Frame 562: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 48942, Seq: 6344, Ack: 6658, Len: 66
▼ OpenFlow 1.0
  .000 0001 = Version: 1.0 (0x01)
  Type: OFPT_PACKET_OUT (13)
  Length: 66
  Transaction ID: 0
  Buffer Id: 0xffffffff
  In port: 1
  Actions length: 8
  Actions type: Output to switch port (0)
  Action length: 8
  Output port: 65531
  Max length: 0
▶ Ethernet II, Src: 00:00:00_00:00:11 (00:00:00:00:00:11), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: 00:00:00_00:00:11 (00:00:00:00:00:11)
  Sender IP address: 10.0.0.1
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.0.0.2

```

Figura 1.4: Respuesta del controlador a ap1

Cuando el paquete llega a ap2 a través de ap1, ocurre lo mismo; ap2 envía una consulta al controlador mediante un mensaje OFPT\_PACKET\_IN para saber qué hacer con el paquete enviado por sta1 a la dirección de broadcast, que ha recibido por su interfaz ap2-mp2. El controlador contesta con un paquete OFPT\_PACKET\_OUT que debe expulsar el paquete por el resto de sus puertos.

Hay que destacar que en estos casos el controlador no indica a ap1 ni a ap2 que instalen entradas nuevas para este tipo de tráfico en sus tablas de flujos, sino que únicamente les indica qué debe hacer con ellos. Si se volviera a enviar un paquete desde sta1 dirigido a la dirección de broadcast, los *access points* tendrían que volver a consultar al controlador.

Sin embargo, en la respuesta de sta2 al ARP request procedente desde sta1, ocurre algo diferente. sta2 responde con un ARP reply, que llega a ap2, y este realiza la pertinente consulta al controlador mostrada en la Figura 1.5.

567	93.3...	00:00:00_00:00:12	00:00:00_00:00...	OpenFlow	126	Type: OFPT_PACKET_IN
568	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
569	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6538 Ack=6368
570	93.3...	00:00:00_00:00:12	00:00:00_00:00...	OpenFlow	132	Type: OFPT_PACKET_OUT
571	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6538 Ack=6434
572	93.3...	00:00:00_00:00:12	00:00:00_00:00...	OpenFlow	126	Type: OFPT_PACKET_IN
573	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD

▶ Frame 567: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0  
 ▶ Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 48944, Dst Port: 6653, Seq: 6478, Ack: 6288, Len: 60  
 ▼ OpenFlow 1.0  
   .000 0001 = Version: 1.0 (0x01)  
   Type: OFPT\_PACKET\_IN (10)  
   Length: 60  
   Transaction ID: 0  
   Buffer Id: 0xffffffff  
   Total length: 42  
   In port: 1  
   Reason: No matching flow (table-miss flow entry) (0)  
   Pad: 00  
 ▶ Ethernet II, Src: 00:00:00\_00:00:12 (00:00:00:00:00:12), Dst: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
 ▼ Address Resolution Protocol (reply)  
   Hardware type: Ethernet (1)  
   Protocol type: IPv4 (0x0800)  
   Hardware size: 6  
   Protocol size: 4  
   Opcode: reply (2)  
   Sender MAC address: 00:00:00\_00:00:12 (00:00:00:00:00:12)  
   Sender IP address: 10.0.0.2  
   Target MAC address: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
   Target IP address: 10.0.0.1

Figura 1.5: Consulta de ap2 al controlador

En este caso el controlador envía dos paquetes como respuesta a la consulta de ap2. Se trata de un mensaje OFPT\_FLOW\_MOD, en el que indica a ap2 que debe añadir una nueva entrada a su tabla de flujos para el tráfico que se indica en el mensaje OFPT\_PACKET\_OUT. Las Figuras 1.6 y 1.7 muestran estos dos paquetes capturados.

Por tanto, ap2 añade un nuevo flujo en su tabla para el tráfico ARP reply procedente de sta2 y destinado a sta1, que entra por el puerto 1 de ap2, correspondiente a la interfaz 'ap2-wlan1', para que sea enviado por el puerto 3 de ap2, interfaz 'ap2-mp2'. En ap1 ocurre lo mismo: tras su consulta al controlador para ese tipo de tráfico, este le responde con dos mensajes para que instale un nuevo flujo en su tabla.

Se puede comprobar que realmente estos flujos se instalan en los *access points*: mirando el listado de flujos del apartado 1.4 se deben encontrar estas reglas.

En ap1:

```
cookie=0x0, duration=5.054s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, arp, in_port='ap1-mp2',vlan_tci=0x0000,d1_src=00:00:00:00:00:12,d1_dst=00:00:00:00:00:11, arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions=output:'ap1-wlan1'
```

568	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
569	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6653
570	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	132	Type: OFPT_PACKET_OUT
571	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6653
572	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	126	Type: OFPT_PACKET_IN
573	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
574	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6653
575	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	132	Type: OFPT_PACKET_OUT
576	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6653
577	93.3...	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN
578	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
579	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6653

▶ Frame 568: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0  
 ▶ Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 48944, Seq: 6288, Ack: 6538, Len: 146  
 ▼ OpenFlow 1.0  
 .000 0001 = Version: 1.0 (0x01)  
 Type: OFPT\_FLOW\_MOD (14)  
 Length: 80  
 Transaction ID: 0  
 Wildcards: 0  
 In port: 1  
 Ethernet source address: 00:00:00\_00:00:12 (00:00:00:00:00:12)  
 Ethernet destination address: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
 Input VLAN id: 65535  
 Input VLAN priority: 0  
 Pad: 00  
 DL type: 2054  
 IP ToS: 0  
 IP protocol: 2  
 Pad: 0000  
 Source Address: 10.0.0.2  
 Destination Address: 10.0.0.1

Figura 1.6: Respuesta de modificación del controlador a ap2

570	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	132	Type: OFPT_PACKET_OUT
571	93.3...	127.0.0.1	127.0.0.1	TCP	66	48944 → 6653 [ACK] Seq=6538 Ack=6434
572	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	126	Type: OFPT_PACKET_IN
573	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
574	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6718 Ack=6490
575	93.3...	00:00:00_00:00:12	00:00:00_00:00:11	OpenFlow	132	Type: OFPT_PACKET_OUT
576	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6718 Ack=6556
577	93.3...	10.0.0.1	10.0.0.2	OpenFlow	182	Type: OFPT_PACKET_IN
578	93.3...	127.0.0.1	127.0.0.1	OpenFlow	146	Type: OFPT_FLOW_MOD
579	93.3...	127.0.0.1	127.0.0.1	TCP	66	48942 → 6653 [ACK] Seq=6834 Ack=6636

▶ Frame 570: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 0  
 ▶ Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 6653, Dst Port: 48944, Seq: 6368, Ack: 6538, Len: 66  
 ▼ OpenFlow 1.0  
 .000 0001 = Version: 1.0 (0x01)  
 Type: OFPT\_PACKET\_OUT (13)  
 Length: 66  
 Transaction ID: 0  
 Buffer Id: 0xffffffff  
 In port: 1  
 Actions length: 8  
 Actions type: Output to switch port (0)  
 Action length: 8  
 Output port: 3  
 Max length: 0  
 ▶ Ethernet II, Src: 00:00:00\_00:00:12 (00:00:00:00:00:12), Dst: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
 ▼ Address Resolution Protocol (reply)  
 Hardware type: Ethernet (1)  
 Protocol type: IPv4 (0x0800)  
 Hardware size: 6  
 Protocol size: 4  
 Opcode: reply (2)  
 Sender MAC address: 00:00:00\_00:00:12 (00:00:00:00:00:12)  
 Sender IP address: 10.0.0.2  
 Target MAC address: 00:00:00\_00:00:11 (00:00:00:00:00:11)  
 Target IP address: 10.0.0.1

Figura 1.7: Respuesta del controlador con las acciones a instalar en ap2



En ap2:

```
cookie=0x0, duration=6.802s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=65535, arp,in_port='ap2-wlan1',vlan_tci=0x0000,dl_src=00:00:00:00:00:12,dl_dst=00:00:00:00:00:11, arp_spa=10.0.0.2,arp_tpa=10.0.0.1,arp_op=2 actions=output:'ap2-mp2'
```

De igual manera, cuando sta1 envía el ICMP echo request hacia sta2, al llegar el paquete a la interfaz 'ap1-wlan1', ap1 consultará al controlador, como muestra la figura 1.8.

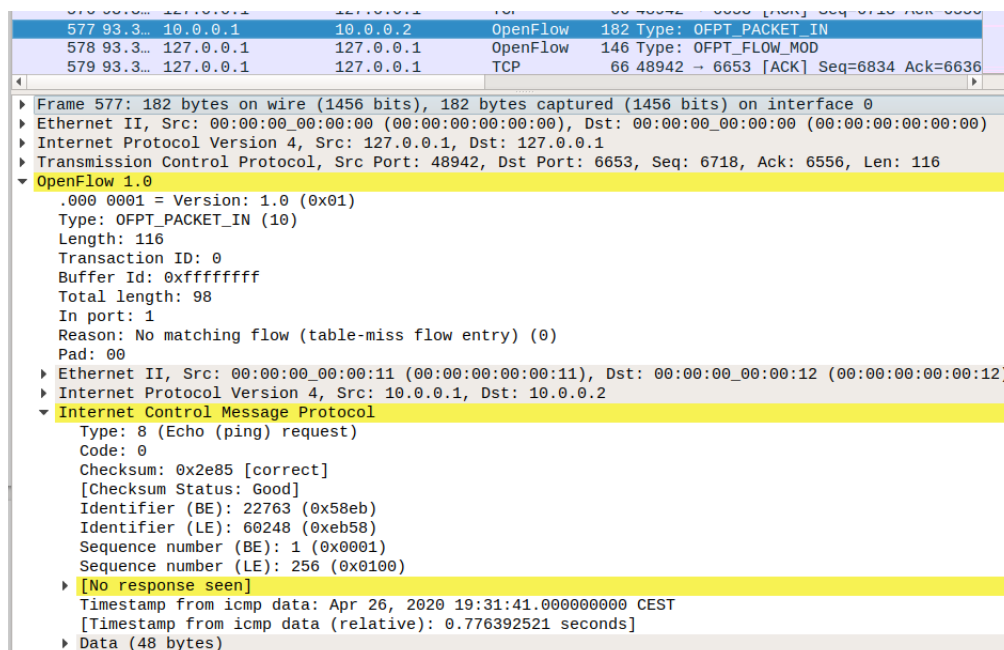


Figura 1.8: Consulta de ap1 del ICMP echo request

Igual que en el caso anterior, el controlador contesta a ap1 indicándole que debe instalar un nuevo flujo para el tráfico ICMP echo request procedente de sta1 hacia sta2, que entra por el puerto 1, interfaz 'ap1-wlan1' y que debe ser enviado por el puerto 3 'ap1-mp2'. En las Figuras 1.9 y 1.10 se muestran estos mensajes.

Los restantes flujos instalados en los *access points* se añaden a las tablas de flujos de igual manera que los dos últimos casos analizados.

No obstante, a parte de indicar cómo es la instalación de estos flujos, las capturas también aportan información acerca de las interfaces que se conectan entre ellas:

- En sta1 'sta1-wlan1' se conecta con 'ap1-wlan1'.

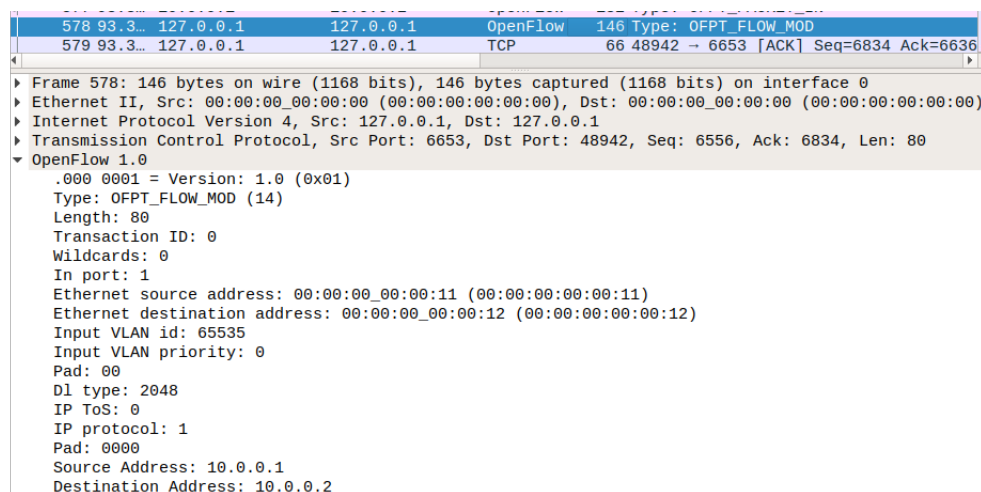


Figura 1.9: Respuesta del controlador con la modificación del flujo en ap2

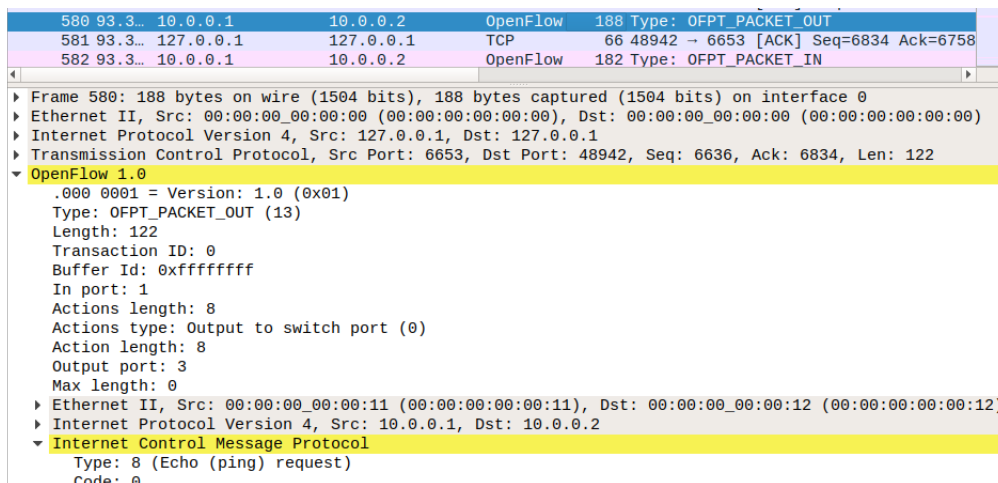


Figura 1.10: Respuesta del controlador con las acciones a instalar en ap1

- En ap1, a parte de la conexión indicada anteriormente, 'ap1-mp2' se conecta con 'ap2-mp2'.
- En ap2, además de la conexión con ap1, 'ap2-wlan1' se conecta con 'sta2-wlan1'.

Con esta información de las conexiones entre los nodos es sencillo completar el esquema de la topología de la red mostrada en en el apartado 1.2. La topología resultante se muestra en el apartado 1.6.

## 1.6. Topología completa resultante

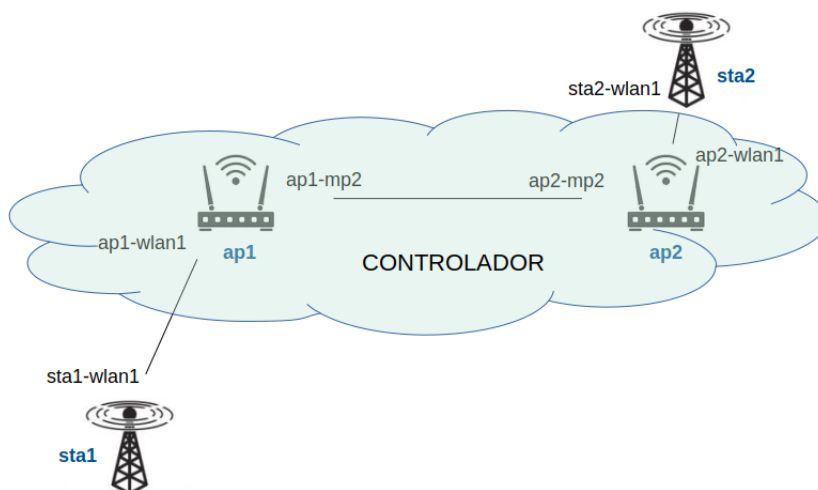


Figura 1.11: Topología resultante de la red meshAP.