

- f) ¿Por qué no hay ningún paquete ICMP en la captura, a pesar de provenir de la ejecución de un “ping” entre dos hosts? (0,1 puntos)

Porque h2 nunca recibe respuesta a su petición ARP, con lo que no sabe qué dirección MAC tiene h3 y por tanto no puede encapsular la petición de eco en una trama Ethernet que vaya destinada a h3.

- g) ¿Estaba activada la aplicación “proxy ARP” en el controlador ONOS cuando se realizó este experimento? Razona su respuesta. (0,1 puntos)

No, ya que, de haberlo estado, la consulta ARP realizada por h2 habría recibido la correspondiente respuesta con la dirección MAC de h3, y se habría observado más tráfico en la captura.

A continuación se muestran los flujos de uno de los dispositivos del escenario tal y como se visualiza con ONOS:

TATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	7	417	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	87	2,970	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	259	265	10	0	IN_PORT:2, ETH_DST:00:00:00:00:00:02, ETH_SRC:00:00:00:00:00:03	imm[OUTPUT:3], cleared:false	*fwd
Added	259	265	10	0	IN_PORT:3, ETH_DST:00:00:00:00:00:03, ETH_SRC:00:00:00:00:00:02	imm[OUTPUT:1], cleared:false	*fwd
Added	1,917	2,970	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1,917	2,970	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

- h) Si este dispositivo recibe un paquete IP por su puerto 2, encapsulado en una trama Ethernet con direcciones de destino = 00:00:00:00:00:02, y de origen = 00:00:00:00:00:03, ¿qué entrada o entradas de la tabla anterior utilizará el dispositivo para dar el tratamiento correspondiente a ese paquete? ¿Por qué? ¿Qué tratamiento le dará a dicho paquete? (0,2 puntos)

Utilizará la tercera entrada, siendo el tratamiento (TREATMENT) enviar el paquete por el puerto 3 (imm[OUTUPUT:3]). Por qué: el paquete “encaja” con los selectores del flujo de la primera (por ser IP) y la tercera (por el puerto de entrada y las direcciones Ethernet de origen y destino) entradas. La tercera tiene prioridad mayor, por lo que es la que se aplicará.

 DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA ETSIS TELECOMUNICACIÓN UPM	REDES Y SERVICIOS AVANZADOS Examen de evaluación continua EC2. 27 de mayo de 2022	
APELLOS: SOLUCIÓN		
NOMBRE:	DNI:	

- i) Si este dispositivo recibe un paquete IP por su puerto 2, encapsulado en una trama Ethernet con direcciones de destino = 00:00:00:00:00:04, y de origen = 00:00:00:00:00:03, ¿qué entrada o entradas de la tabla anterior utilizará el dispositivo para dar el tratamiento correspondiente a ese paquete? ¿Por qué? ¿Qué tratamiento le dará a dicho paquete? (0,2 puntos)

Utilizará la primera entrada, siendo el tratamiento enviar la trama Ethernet recibida dentro de un mensaje PACKET_IN al controlador ("imm[OUTPUT:CONTROLLER]"). Por qué: en este caso el paquete únicamente encaja en el selector de la primera entrada (por ser IP). Ninguna otra entrada encaja (bien por el protocolo encapsulado en Ethernet, bien por puerto de entrada y direcciones MAC).

A continuación se muestra un mensaje Openflow relacionado con la instalación de un flujo en una tabla de uno de los dispositivos:

```

OpenFlow 1.4
Version: 1.4 (0x05)
Type: OFPT_FLOW_MOD (14)
Length: 104
Transaction ID: 4194559
Cookie: 0x000480000a4e53dad
Cookie mask: 0xffffffffffffffff00000000
Table ID: 0
Command: OFPFC_ADD (0)
Idle timeout: 0
Hard timeout: 0
Priority: 10
Buffer ID: OFP_NO_BUFFER (4294967295)
Out port: OFPP_ANY (4294967295)
Out group: OFPG_ANY (4294967295)
Flags: 0x0001
..... .... .... ...1 = Send flow removed: True
..... .... .... ..0. = Check overlap: False
..... .... .... .0.. = Reset counts: False
..... .... .... 0... = Don't count packets: False
..... .... ....0 .... = Don't count bytes: False
Importance: 0
Match
  Type: OFPMT_OXM (1)
  Length: 32
  OXM field
    Class: OFPXML_OPENFLOW_BASIC (0x8000)
    0000 000. = Field: OFPXMT_OFB_IN_PORT (0)
    ..... 0 = Has mask: False
    Length: 4
    Value: 1
  OXM field
    Class: OFPXML_OPENFLOW_BASIC (0x8000)
    0000 011. = Field: OFPXMT_OFB_ETH_DST (3)
    ..... 0 = Has mask: False
    Length: 6
    Value: 00:00:00_00:00:04 (00:00:00:00:00:04)

```

(Continúa en página siguiente)

```

OXM field
  Class: OFPXM_C_OPENFLOW_BASIC (0x8000)
  0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
  .... ...0 = Has mask: False
  Length: 6
  Value: 00:00:00_00:00:01 (00:00:00:00:00:01)

Instruction
  Type: OFPIT_APPLY_ACTIONS (4)
  Length: 24
  Pad: 00000000

Action
  Type: OFPAT_OUTPUT (0)
  Length: 16
  Port: 4
  Max length: 0
  Pad: 000000000000

```

- j) Rellene los valores del flujo que se crea mediante el mensaje anterior en la siguiente tabla (0,2 puntos):

FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT
10	0	IN_PORT:1, ETH_DST:00:00:00:00:00:04, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:4], cleared:false

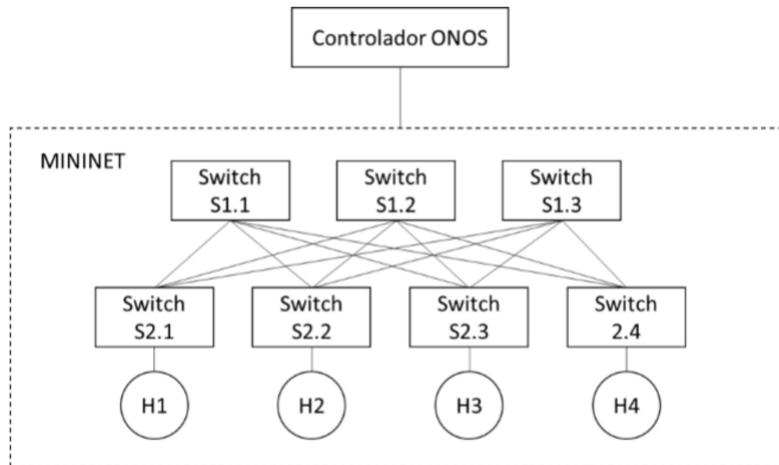
APELLIDOS:

NOMBRE:

DNI:

EJERCICIO 6. Puntuación: 1,5 puntos. Tiempo estimado: 15 minutos

En el siguiente esquema de red se muestra una topología de switches y hosts ejecutados dentro del entorno Mininet y conectados a un controlador ONOS.



Existen dos niveles de switches: el nivel 1 (que incluye a los switches con nombre S1.X) que tiene funcionalidad similar a los switches Spine de la práctica 3 ("Configuración de una red SDN y análisis de tráfico OpenFlow"), y el nivel 2 (switches con nombre S2.X) que corresponden a los switches de tipo Leaf en esa misma práctica. Los switches no tienen ningún tipo de inteligencia (entorno SDN puro en las mismas condiciones que en la práctica de SDN de la asignatura) y son compatibles con el protocolo OpenFlow. El controlador también es compatible con el protocolo OpenFlow.

El entorno de despliegue es el mismo que en la práctica 3 de la asignatura. Se han configurado las direcciones IP, máscaras y direcciones MAC según la siguiente tabla:

Equipo	Dirección IP / máscara	Dirección MAC (último octeto)
Controlador	10.10.0.1/30	-
Mininet	10.10.0.2/30	-
H1	10.10.1.1/24	0A
H2	10.10.1.2/24	0B
H3	10.10.1.3/24	0C
H4	10.10.1.4/24	0D

Tras el arranque de Mininet, se ha visto que se han asignado los siguientes puertos TCP a cada uno de los switches (interfaz Mininet-Controlador):

Switch	Puerto TCP
S1.1	10001
S1.2	10002
S1.3	10003
S2.1	20001
S2.2	20002
S2.3	20003
S2.4	20004

Todos los switches tienen cuatro puertos de entrada/salida. Cada puerto se numera del 1 al 4 y está conectado a un equipo concreto que se describe en la siguiente tabla:

Equipo	Puerto	Conectado al equipo
S1.1	$1 \leq X \leq 4$	S2.X
S1.2	$1 \leq X \leq 4$	S2.X
S1.3	$1 \leq X \leq 4$	S2.X
S2.1	$1 \leq X \leq 3$	S1.X
S2.1	4	H1
S2.2	$1 \leq X \leq 3$	S1.X
S2.2	4	H2
S2.3	$1 \leq X \leq 3$	S1.X
S2.3	4	H3
S2.4	$1 \leq X \leq 3$	S1.X
S2.4	4	H4

De la tabla anterior se puede deducir que, por ejemplo, el puerto 2 ($X=2$) del switch S1.1 estará conectado al equipo S2.2 (de nuevo $X=2$). Otro ejemplo es que el puerto 1 ($X=1$) de S2.3 estará conectado al equipo S1.1. Finalmente, el puerto número 4 de todos los equipos de segundo nivel (S2.1, S2.2, S2.3 y S2.4) están conectados a sus respectivos hosts.

Al igual que en la práctica, se pueden monitorizar todas las interfaces, tanto las involucradas en comunicaciones OpenFlow como las internas de Mininet. Además, están disponibles todas las aplicaciones del controlador ONOS utilizadas en la práctica, con el mismo comportamiento, aunque adaptadas al cambio de topología. Con la información suministrada, indique si las aserciones realizadas en cada uno de los escenarios son verdaderas o falsas (+0,1 por respuesta correcta y -0,1 por respuesta incorrecta).

Escenario 1: Se despliega la red Mininet de forma satisfactoria, el controlador tiene activas las aplicaciones “org.onosproject.proxyarp” y “org.onosproject.fwd” estando el proceso de captura de paquetes activo en todas las interfaces. A continuación, se ejecuta el siguiente mandato en Mininet sin haber ejecutado ningún otro mandato anteriormente: “h1 ping -c1 h4”.

- V Se capturan un total de catorce mensajes del tipo OFPT_HELLO.
- F El controlador asignará un total de 4 “datapath_id”: uno por cada switch de la capa más baja, es decir, de tipo Leaf, ya que son los que tienen host conectados directamente.
- F Se capturarán un total de tres paquetes OFPT_PACKET_OUT con puerto TCP origen 6653.
- F No se capturará ningún paquete OFPT_PACKET_IN con puerto origen 20003.
- F Los tres switches de la capa más alta (S1.1, S1.2 y S1.3) enviarán un OFPT_PACKET_IN al controlador con un “ICMP echo request” encapsulado.
- V El switch S2.1 recibirá al menos un mensaje OFPT_PACKET_OUT con un mensaje “arp” encapsulado.
- V El switch S2.1 enviará al menos un mensaje OFPT_PACKET_IN con un mensaje “arp” encapsulado.
- F El controlador recibirá tres mensajes OFPT_PACKET_IN de los tres switches de la capa más alta (S1.1, S1.2 y S1.3) con un mensaje ARP encapsulado.



DEPARTAMENTO DE INGENIERÍA
TELEMÁTICA Y ELECTRÓNICA
ETSIIS TELECOMUNICACIÓN
UPM

REDES Y SERVICIOS AVANZADOS
Examen convocatoria extraordinaria – Parte EC2 - 12 de julio de 2023

APELLIDOS:

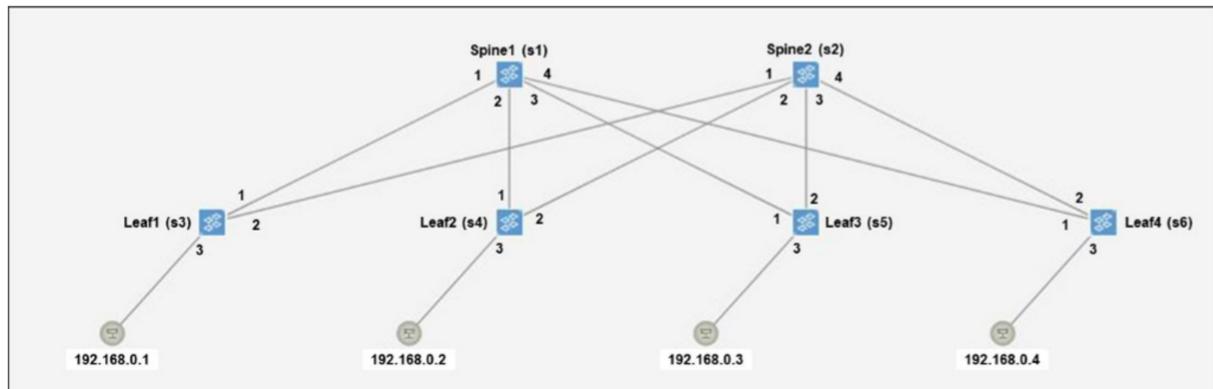
NOMBRE:

DNI:

- F** El controlador enviará un mensaje OFPT_PACKET_OUT a uno de los tres switches de la capa más alta (S1.1, S1.2 y S1.3) con un mensaje ARP encapsulado.
- V** El switch S2.1 recibirá varios mensajes OFPT_FLOW_MOD.
- V** Se capturarán mensajes OFPT_FLOW_MOD con puerto destino 20002.
- F** Se capturarán mensajes OFPT_FLOW_MOD en el puerto 1, 2 o 3 del Switch S2.4.
- F** El controlador enviará un mensaje OFPT_FLOW_MOD a cada uno de los tres switches de la capa más alta (S1.1, S1.2 y S1.3).
- F** El host H1 recibirá un mensaje OFPT_PACKET_OUT con un mensaje "ICMP Echo Reply" encapsulado.
- F** El host H2 no recibe ningún tipo de paquete de datos en ningún momento.

EJERCICIO 7. Puntuación: 1 punto. Tiempo estimado: 10 minutos

En la figura siguiente se muestra la topología de la red "Spine&Leaf" que se utilizará para la resolución de este ejercicio sobre la práctica 3 "Configuración de una red SDN y análisis de tráfico OpenFlow".



En las figuras siguientes se muestra la información proporcionada por el controlador ONOS sobre los hosts y los flujos instalados en uno de los dispositivos de la red ("Leaf4 (s6)").

Hosts (4 total)						
FRIENDLY NAME ▾	HOST ID	MAC ADDRESS	VLAN ID	CONFIGURED	IP ADDRESSES	LOCATION
192.168.0.4	00:00:00:00:D/None	00:00:00:00:0D	None	false	192.168.0.4	of:0000000000000006/3
192.168.0.3	00:00:00:00:0C/None	00:00:00:00:0C	None	false	192.168.0.3	of:0000000000000005/3
192.168.0.2	00:00:00:00:0B/None	00:00:00:00:0B	None	false	192.168.0.2	of:0000000000000004/3
192.168.0.1	00:00:00:00:0A/None	00:00:00:00:0A	None	false	192.168.0.1	of:0000000000000003/3

Flows for Device of:0000000000000006 (5 Total)							
STATE ▾	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	200	55	0	IN_PORT:3, ETH_DST:00:00:00:00:00:0C, ETH_SRC:00:00:00:00:00:0D	imm[OUTPUT:2], cleared:false	*net.intent
Added	0	200	55	0	IN_PORT:2, ETH_DST:00:00:00:00:00:0D, ETH_SRC:00:00:00:00:00:0C	imm[OUTPUT:3], cleared:false	*net.intent
Added	4	900	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	376	900	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	376	900	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

APELLIDOS:

NOMBRE:

DNI:

Teniendo en cuenta la información anterior, se pide:

- a) Indique razonadamente las configuraciones que han sido realizadas en el controlador de ONOS para obtener el resultado mostrado en el recuadro siguiente con la ejecución del comando “pingall” en Mininet. Especifique, además, todos los modos posibles para realizar dicha configuración. (0,3 puntos)

```

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X
h2 -> h1 X X
h3 -> X X h4
h4 -> X X h3
*** Results: 66% dropped (4/12 received)
mininet>

```

Analizando el resultado del comando “pingall” de Mininet, se ve que hay conectividad entre “h1-h2” y entre “h3-h4”.

La aplicación “Reactive Forwarding” no está activada, ya que si lo estuviera aparecería en la tabla de flujos una entrada para el tratamiento del tráfico “ETH_TYPE:ipv4”. Por lo tanto, se tienen configurados dos “host-to-host intent”, uno entre h1 y h2 y otro entre h3 y h4 para permitir el tráfico de datos entre dichos hosts.

Un “host-to-host intent” se puede configurar de las tres maneras siguientes:

- Utilizando la interfaz de comandos de ONOS (ONOS CLI).
- Utilizando la interfaz gráfica de ONOS (ONOS GUI).
- Mediante ONOS REST API

- b) En la figura siguiente se muestran los flujos instalados en la tabla de flujos del switch “Spine1 (s1)” una vez realizada la configuración del apartado a). Complete la tabla siguiente con todos los flujos que tendría instalados el switch “Spine2 (s2)” para que se produzca la salida del comando “pingall” de Mininet mostrada en el recuadro del apartado anterior. (0,4 puntos)

Flows for Device of:00000000000000000001 (5 Total)							
STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	217	100	0	IN_PORT:2, ETH_DST:00:00:00:00:0A, ETH_SRC:00:00:00:00:0B	imm[OUTPUT:1], cleared:false	*net.intent
Added	0	217	100	0	IN_PORT:1, ETH_DST:00:00:00:00:0B, ETH_SRC:00:00:00:00:0A	imm[OUTPUT:2], cleared:false	*net.intent
Added	0	1,432	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	3,847	2,980	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	3,847	2,980	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core

SELECTOR	TREATMENT	APP NAME
IN_PORT: 4 ETH_DST:00:00:00:00:00:0C ETH_SRC:00:00:00:00:00:0D	imm[OUTPUT:3], CLEARED:false	*net.intent
IN_PORT: 3 ETH_DST:00:00:00:00:00:0D ETH_SRC:00:00:00:00:00:0C	imm[OUTPUT:4], CLEARED:false	*net.intent
ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], CLEARED:true	*core
ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], CLEARED:true	*core
ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], CLEARED:true	*core

- c) Como consecuencia de la invocación del método REST API para la creación de un “Intent” entre “h3” y “h4”, el controlador ONOS envía el siguiente mensaje del protocolo OpenFlow al dispositivo “Leaf4 (s6)”. Rellene los cuadros en blanco de dicho mensaje con los valores adecuados para la instalación del primero de los flujos mostrado en la figura con la tabla de flujos de dicho dispositivo (Flows for Device of:0000000000000006) al comienzo del ejercicio. (0,3 puntos)

<pre> OpenFlow 1.4 Version: 1.4 (0x05) Type: OFPT_FLOW_MOD Length: 104 Transaction ID: 25165935 Cookie: 0x00ac0000e518f313 Cookie mask: 0x0000000000000000 Table ID: 0 Command: OFPFC_ADD Idle timeout: 0 Hard timeout: 0 Priority: 55 Buffer ID: OFP_NO_BUFFER (4294967295) Out port: OFPP_ANY (4294967295) Out group: OFPG_ANY (4294967295) Flags: 0x0001 Importance: 0 Match Type: OFPMT_OXM (1) Length: 32 OXM field Class: OFPXMC_OPENFLOW_BASIC (0x8000) 0000 000. = Field: OFPXMT_OFB_ETH_DST (3) 0 = Has mask: False Length: 6 Value: 00:00:00:00:00:0C OXM field Class: OFPXMC_OPENFLOW_BASIC (0x8000) 0000 100. = Field: OFPXMT_OFB_ETH_SRC (4) 0 = Has mask: False Length: 6 Value: 00:00:00:00:00:0D Instruction Type: OFPIT_APPLY_ACTIONS (4) Length: 24 Pad: 00000000 Action Type: OFPAT_OUTPUT (0) Length: 16 Port: 2 Max length: 0 Pad: 000000000000 </pre> <p>(Sigue en cuadro derecho)</p>	<p>OXM field</p> <p>Class: OFPXMC_OPENFLOW_BASIC (0x8000) 0000 011. = Field: OFPXMT_OFB_ETH_DST (3)0 = Has mask: False Length: 6 Value: 00:00:00:00:00:0C</p> <p>OXM field</p> <p>Class: OFPXMC_OPENFLOW_BASIC (0x8000) 0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)0 = Has mask: False Length: 6 Value: 00:00:00:00:00:0D</p> <p>Instruction</p> <p>Type: OFPIT_APPLY_ACTIONS (4) Length: 24 Pad: 00000000</p> <p>Action</p> <p>Type: OFPAT_OUTPUT (0) Length: 16 Port: 2 Max length: 0 Pad: 000000000000</p>
--	---

 DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA ETSIS TELECOMUNICACIÓN UPM	REDES Y SERVICIOS AVANZADOS Examen de convocatoria extraordinaria - Parte EC2. 30 de junio de 2022	
	APELLOS:	
	NOMBRE:	DNI:

EJERCICIO 4. Puntuación: 2,5 puntos. Tiempo estimado: 25 minutos

(Nota: en este ejercicio las justificaciones correctas y detalladas, incluyendo cuando sean pertinentes los datos concretos proporcionados en el enunciado, son imprescindibles para obtener la puntuación máxima en cada apartado)

Las siguientes preguntas están referidas a la práctica de laboratorio “Configuración de una red SDN y análisis de tráfico OpenFlow”.

- a) ¿Por qué hay peticiones y respuestas Openflow para las que se utilizan mensajes OFPT_MULTIPART_REQUEST / REPLY? Ponga un ejemplo concreto de petición / respuesta que utilice este tipo de mensajes. ¿De qué manera se sabe qué tipo concreto de petición / respuesta contiene una pareja concreta de mensajes OFPT_MULTIPART_REQUEST / REPLY? (0,5 puntos)

OFPT_MULTIPART_REQUEST / REPLY son mensajes preparados para llevar información Openflow potencialmente voluminosa y que pudiera requerir de más de un mensaje Openflow por exceder su tamaño máximo. Un ejemplo de petición / respuesta es la relacionada con la solicitud del controlador a un switch de información acerca de los puertos que tiene y sus características. La manera de saber el tipo concreto es mediante un campo "Type" como parte de la información que llevan estos mensajes (ej. OFPMP_PORT_DESC).

A continuación, se muestran los flujos de uno de los dispositivos del escenario tal y como se visualiza con ONOS:

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	212	100	0	IN_PORT:3, ETH_DST:00:00:00:00:00:01, ETH_SRC:00:00:00:00:00:03	imm[OUTPUT:1], cleared:false	*net.intent
Added	0	212	100	0	IN_PORT:1, ETH_DST:00:00:00:00:00:03, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:3], cleared:false	*net.intent
Added	21	1,845	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	65	1,845	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	192	197	10	0	IN_PORT:2, ETH_DST:00:00:00:00:00:01, ETH_SRC:00:00:00:00:00:02	imm[OUTPUT:1], cleared:false	*fwd
Added	192	197	10	0	IN_PORT:1, ETH_DST:00:00:00:00:00:02, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:2], cleared:false	*fwd
Added	2,386	1,845	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,386	1,845	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core

Uno de estos flujos se creó como consecuencia directa e inmediata de la ejecución de esta orden en la interfaz CLI de ONOS:

```
karaf@root onos> app activate org.onosproject.fwd
```

- b) ¿Qué flujo es? ¿Por qué se crea dicho flujo al activar esa aplicación en ONOS? Indique claramente cuál es su propósito y cómo se relaciona con el funcionamiento de la aplicación. (0,5 puntos)

El tercer flujo. Este flujo se crea para que cuando el switch reciba una trama Ethernet que lleve encapsulada un datagrama IPv4, la encapsule en un mensaje PACKET_IN de Openflow y se la envíe al controlador, quien la entregará a la aplicación "Reactive Forwarding". La aplicación, con la información sobre el datagrama recibido por el switch y el puerto de entrada, y teniendo en cuenta la topología de la red, decidirá qué acciones llevar a cabo (enviar mediante PACKET_OUT el datagrama por el puerto de salida del switch que corresponda e instalar en la tabla de flujos del switch los flujos relacionados con el camino del tráfico IP con mismos origen y destino.

Dos de los flujos se crearon como consecuencia directa y casi inmediata de la ejecución de esta orden en la interfaz CLI de Mininet (se omiten mediante "X" e "Y" los números concretos de host involucrados):

```
mininet> hX ping hY
```

- c) ¿Qué dos flujos son? ¿Qué aplicación provoca que se creen estos flujos y con qué propósito? ¿Cómo se compara la prioridad de estos dos flujos con la prioridad del flujo mencionado anteriormente y qué utilidad tiene esta diferencia de prioridades en el contexto de la aplicación? Razona sus respuestas. (0,5 puntos)

Los flujos quinto y sexto ya que, al ejecutar el ping, se desencadena el envío de datagramas IP al controlador para cada sentido (solicitud y respuesta de eco) que serán entregados a la aplicación "Reactive Forwarding", quien entre otras cosas instalará estos flujos en el switch. En la columna "APP NAME" se ve cómo son los flujos 5 y 6 los que han sido creados por dicha aplicación ("*fwd"). El propósito de los flujos es evitar que para todo el tráfico IP con mismos orígenes y destinos se tenga que enviar el tráfico al controlador (mediante PACKET_IN), utilizándose estos flujos en lo sucesivo, que indican directamente por dónde reenviar este tráfico. Puesto que la prioridad de estos flujos (10) es mayor que la del flujo inicial (5), el tráfico IP con los mismos interlocutores del ping no se enviará al controlador tras el primer paquete en cada sentido.

 DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA ETSIS TELECOMUNICACIÓN UPM	REDES Y SERVICIOS AVANZADOS Examen de convocatoria extraordinaria - Parte EC2. 30 de junio de 2022	
APELLOS:		
NOMBRE:		DNI:

- d) ¿Cuáles son los primeros mensajes Openflow que se intercambian el switch al que pertenece la tabla anterior y el controlador como consecuencia de la ejecución del ping anterior? Indique los mensajes Openflow que se observarán hasta que los dos flujos de la pregunta anterior están creados, y describa brevemente el propósito y el contenido fundamental de cada uno. No incluya en este análisis los mensajes ARP, es decir, puede suponer que las tablas ARP de los sistemas finales involucrados contienen la información necesaria. (1 punto)

- 1) **PACKET_IN:** Cuando el switch recibe el echo request por uno de sus puertos, utilizando la tercera entrada de la tabla de flujos enviará un **PACKET_IN** al controlador con dicho paquete.
- 2) **PACKET_OUT:** El controlador entregará a la aplicación Reactive Forwarding y esta, calculando el camino entre origen y destino, causará el envío de un **PACKET_OUT** que contiene la petición de eco al switch indicándole que lo envíe por el puerto que corresponda al camino calculado.
- 3) **FLOW_MOD:** La aplicación también causa que el controlador instale el flujo correspondiente al camino elegido entre origen y destino en la tabla de flujos (sería el flujo 5 o el 6, con los datos del enunciado no se puede saber si uno u otro).
- 4) **BARRIER_REQUEST:** Tras el **FLOW_MOD** el controlador se asegura de controlar el orden y el flujo de ejecución de las órdenes enviadas al switch mediante este mensaje, al que el switch contestará con un **BARRIER_REPLY**.
- 5) **PACKET_IN:** Cuando el switch recibe el echo reply por uno de sus puertos, lo envía al controlador por motivos análogos al paso 1)
- 6) **PACKET_OUT:** Lo envía el controlador por motivos análogos al paso 2) (para el tráfico "de vuelta").
- 7) **FLOW_MOD, BARRIER_REQUEST/REPLY:** Motivos análogos al paso 3) y 4), para el tráfico "de vuelta").