

## Part1

Q1:

Ans: 有 6 個

Q2,Q3:

Unknown type 是 BDDP(0x8942)

Match fields	Action	Timeout values
ETH_TYPE=ARP	OUTPUT_PORT=CONTROLLER	0
ETH_TYPE=UNKNOWN(BDDP)	OUTPUT_PORT=CONTROLLER	0
IN_PORT=1 ETH_SRC=52:D6:3C:5B:7D:22(h1 mac address) ETH_DST=52:D1:87:8A:E9:36(h2 mac address)	OUTPUT_PORT=2	10
IN_PORT=2 ETH_SRC=52:D1:87:8A:E9:36(h2 mac address) ETH_DST=52:D6:3C:5B:7D:22(h1 mac address)	OUTPUT_PORT=1	10
ETH_TYPE=IPV4	OUTPUT_PORT=CONTROLLER	0
ETH_TYPE=LLDP	OUTPUT_PORT=CONTROLLER	0

## Part2

安裝 flow rule 後，h1 arping h2 成功

The screenshot shows a Linux desktop environment with a terminal window and the ONOS web interface. The terminal window displays the installation of ONOS components and the execution of the 'demo@SDN-NFV' script. The ONOS web interface shows the 'Flows for Device of:0000000000000001 (6 Total)' table.

**Terminal Output:**

```
demo@SDN-NFV: ~$ ./demo@SDN-NFV
*** Adding links:
(h1, s1) (h2, s1)
*** configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 arping h2
ARPING 10.0.0.2
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=0 time=35.8072 usec
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=1 time=3.200 usec
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=2 time=2.754 usec
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=3 time=4.686 usec
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=4 time=4.196 usec
42 bytes from 46:38:28:aa:2d:da (10.0.0.2): index=5 time=3.479 usec

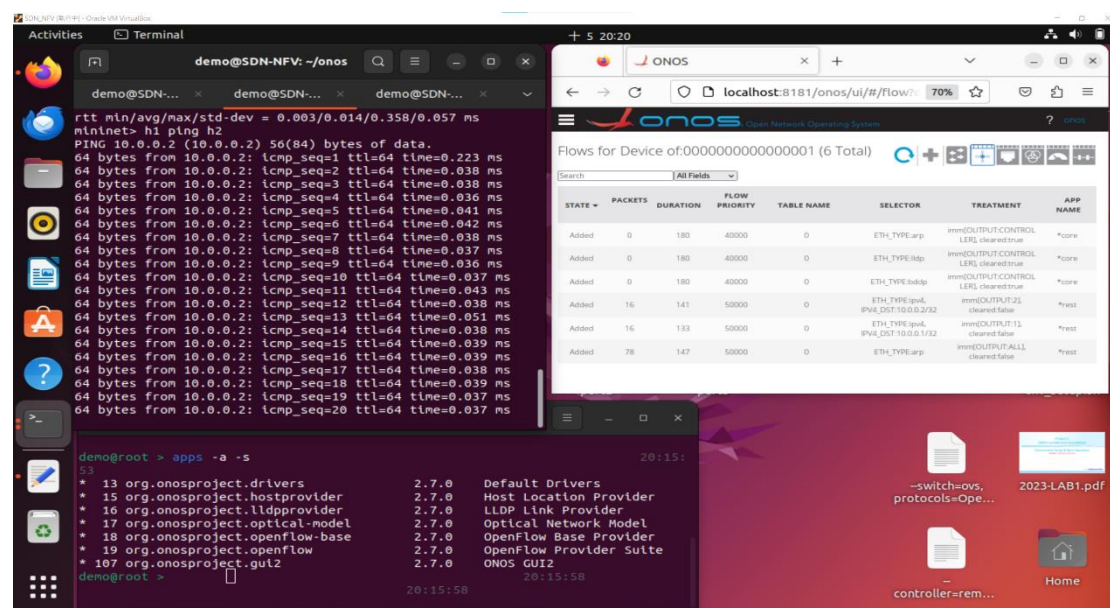
demo@root > apps -a -s
53
* 13 org.onosproject.drivers 2.7.0 Default Drivers
* 15 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 16 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 17 org.onosproject.optical-model 2.7.0 Optical Network Model
* 18 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider
* 19 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite
* 107 org.onosproject.gui2 2.7.0 ONOS GUI2
demo@root >
```

**ONOS Web Interface:**

Flows for Device of:0000000000000001 (6 Total)

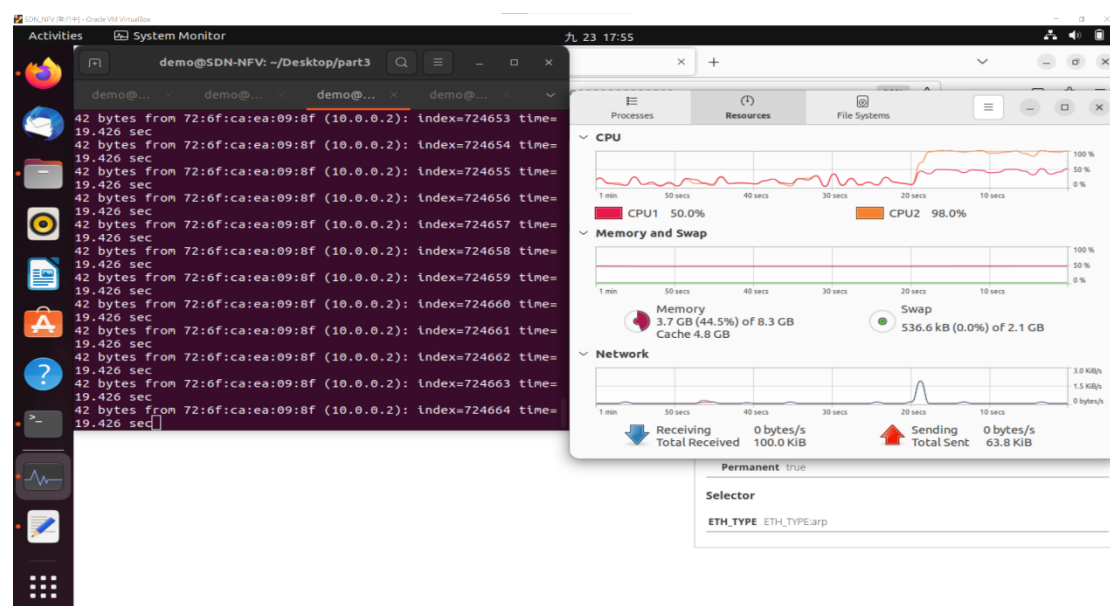
STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	130	40000	0	ETH_TYPE=arp	imm(OUTPUT_CONTROL LER), cleared true	*core
Added	0	91	50000	0	ETH_TYPE=ip4, IPV4_DST=10.0.0.2/32	imm(OUTPUT2), cleared false	*rest
Added	0	130	40000	0	ETH_TYPE=lldp	imm(OUTPUT_CONTROL LER), cleared true	*core
Added	0	83	50000	0	ETH_TYPE=ip4, IPV4_DST=10.0.0.1/32	imm(OUTPUT1), cleared false	*rest
Added	0	130	40000	0	ETH_TYPE=bddp	imm(OUTPUT_CONTROL LER), cleared true	*core
Added	10	97	50000	0	ETH_TYPE=arp	imm(OUTPUT_ALL), cleared false	*rest

安裝 flow rule 後，h1 ping h2 成功



### Part3

create 一個拓樸，拓樸有 3 個 switch 跟 2 個 host，這三個 switch 互相連接成一個環狀，h1 link to s2 and h2 link to s2，為三個 switch install flow rule，這時當 h1 arping h2 時，h1 送 broadcast packet 給 s2，s2 會轉傳給 s1 跟 s3，當 s1 收到 s2 的封包後會再轉傳給 s3，s3 會收到 s2 的封包會轉傳給 h2、s1，s1 受到 s3 的封包會轉重給 s2，s3 收到 s1 的封包會轉傳給 s2，因為一直循環傳送封包給 h2，h2 會一直 reply，可以看到圖裡一直收到 reply，這時在環狀裡的 broadcast 封包會一直複製並傳送，這就形成 broadcast storm，而因為 broadcast packet 是由 CPU 處理，所以可以發現 CPU 的使用率會大幅上升。



### Part4

Step1. 在 data plane 中 h1 send arp request packet 給 s1，在 s1 的 flow rule 中，

預設 arp request packet 會 output 給 controller，所以會把 arp request packet encapsulate 後 packet in 給 controller plane 的 controller

Step2. 在 controll plane 中，送給 controller 後，ReactivePacketProcessor 會執行 process()，因為 dst mac address 為 broadcast address，會 flood 出去，由於 s1 只連接 2 台 host，controller 會 packet out 給 data plane 中的 s1，通知 S1 從 port2 送出去

Step3. 在 data plane 中，h2 收到 arp request packet，會填上自己的 mac address 後回送 arp reply packet 給 s1，s1 的 flow rule 會將 arp reply packet encapsulate 後 pack in 給 controller plane 的 controller

Step4. 在 controll plane 中，送給 controller 後，ReactivePacketProcessor 會執行 process() function，接著進入 installRule() function，在開始判斷是 arp packet，因此直接 packet out 給 data plane 的 S1 從對應的 port 1 送出去，不會 install rule

Step5. data plane 中 h1 透過 arp 得到 h2 的 mac address 後發送 icmp package，package 到達 s1 後，s1 沒找到匹配的 flow rule，將封包 encapsulate 後 packet in 給 controller

Step6. controll plane 中的 controller 收到後 ReactivePacketProcessor 會執行 process() function，接著進入 installRule() function，在 install rule 後 packet out 給 data plane 的 s1 讓封包從 port2 傳出去

Step7. data plane 中的 s1 收到後透過 port2 將 icmp packet 傳給 h2

## What do I learn and solve

這次的作業在透過 wireshark 抓取 openflow 封包有學到很多新的東西，為了看懂每個 type 代表的意思，自行上網查了對應的解釋，對 openflow protocol 有更深入的了解。

在 part2 的 install rule，有上官方 doc 查 json 檔的寫法，但關於某些參數能填的 value 沒有過多的描寫，後來去看 java doc 才找到答案。

在 part4 的 trace code 是我這次作業學到最多的地方，透過 trace code 配上 wireshark 可以清楚知道整個 forwarding 的過程。