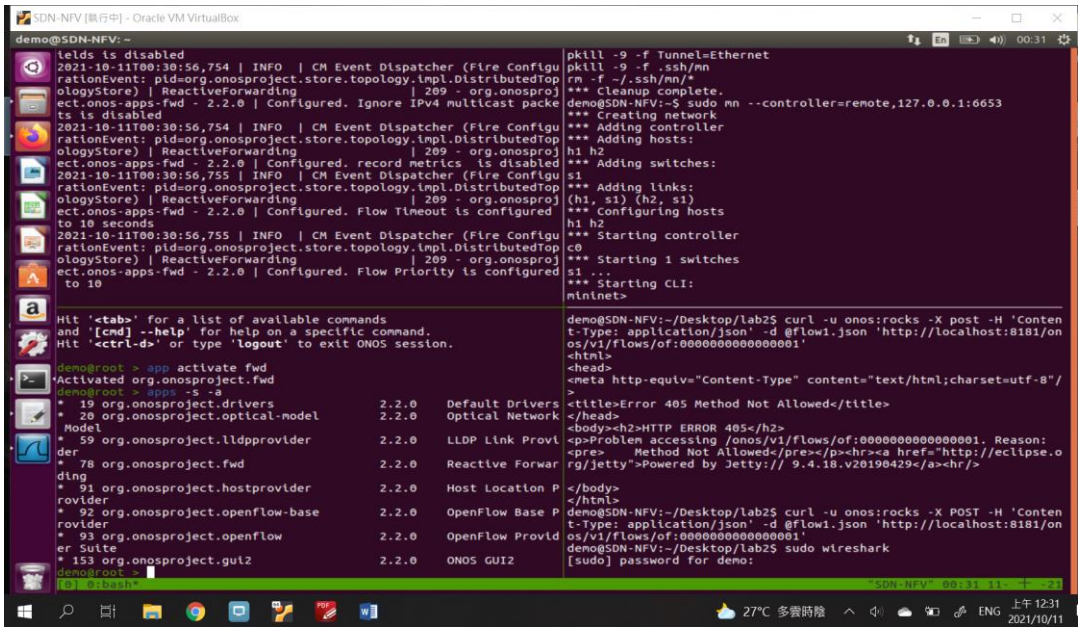
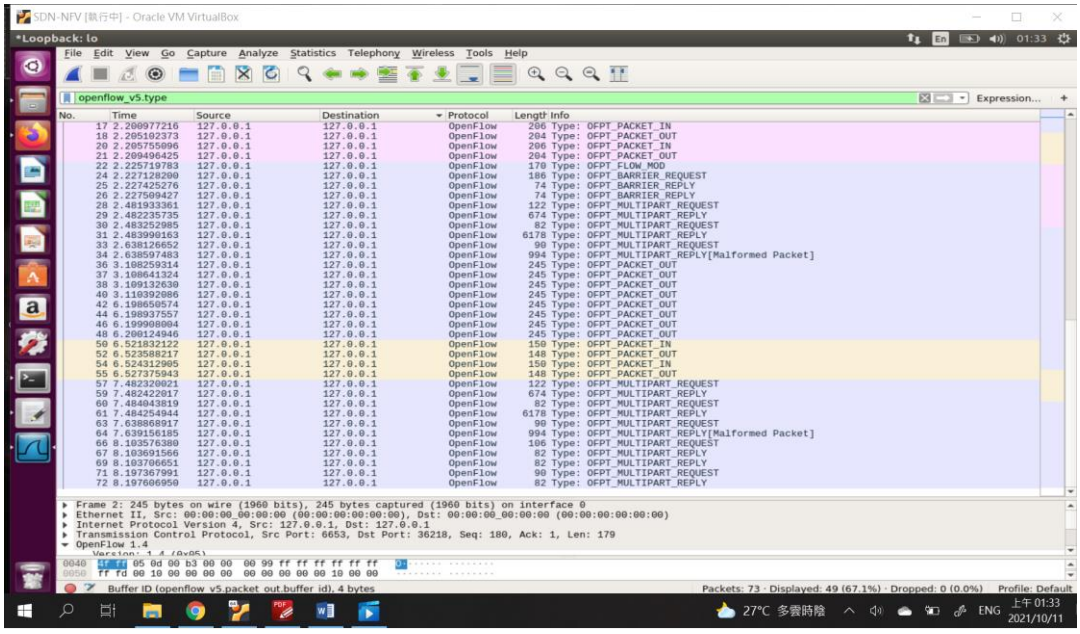
SDN project2 report

Part1

以下在完成 spec 中的 preparation 後的畫面(右上角為 mininet topology，左下角為 onos localhost)

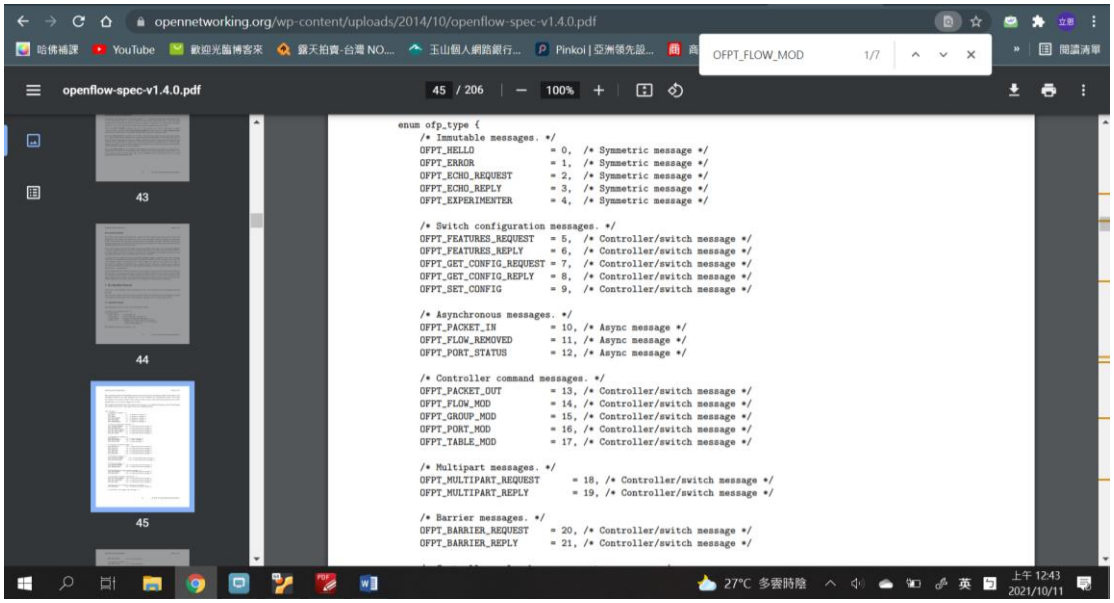


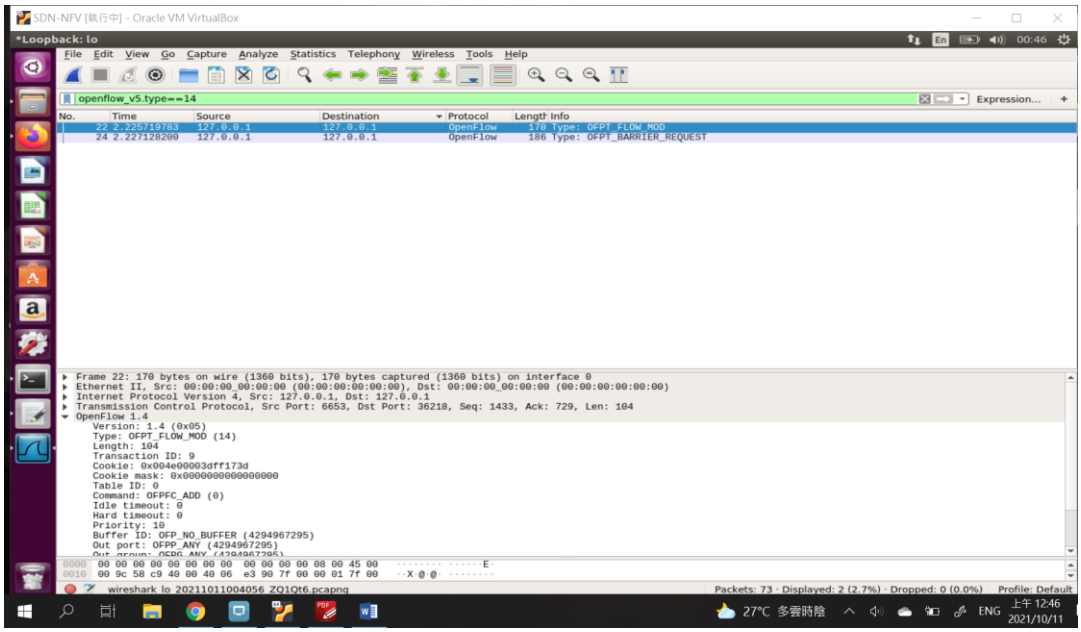
下圖中的 wireshark 介面已經完成對 h1 ping h2 的封包收集



Q1 :

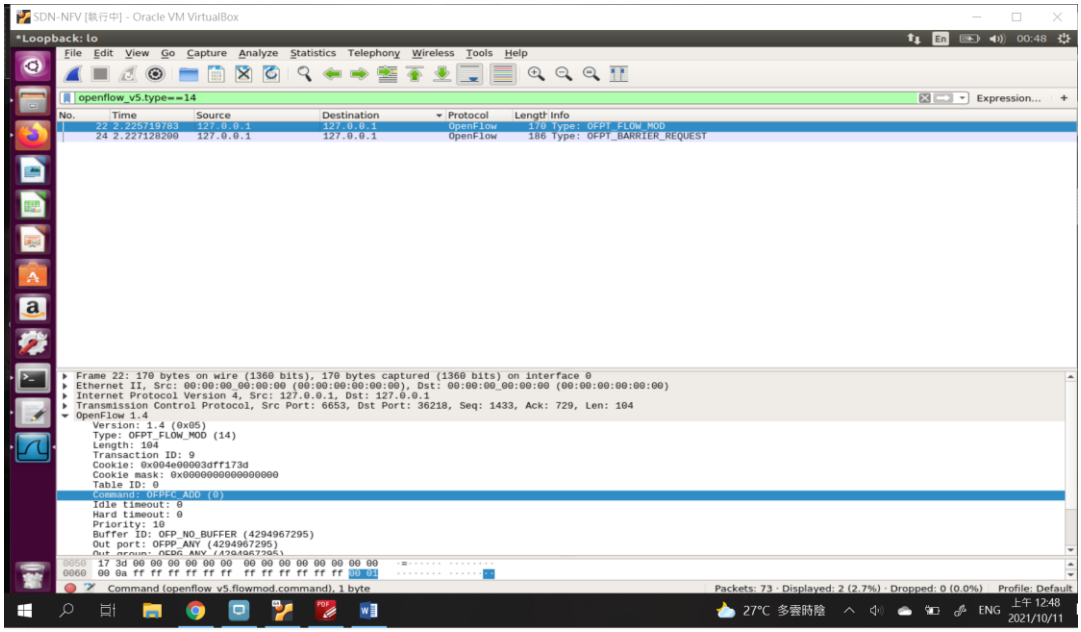
根據 Openflow1.4 版本的 spec，可知 OFPT_FLOW_MOD 的 type 對應到 14，因此用 wireshark 內建的 filter 搜尋 openflow_v5.type==14 的方法尋找含有 OFPT_FLOW_MOD 的 packet



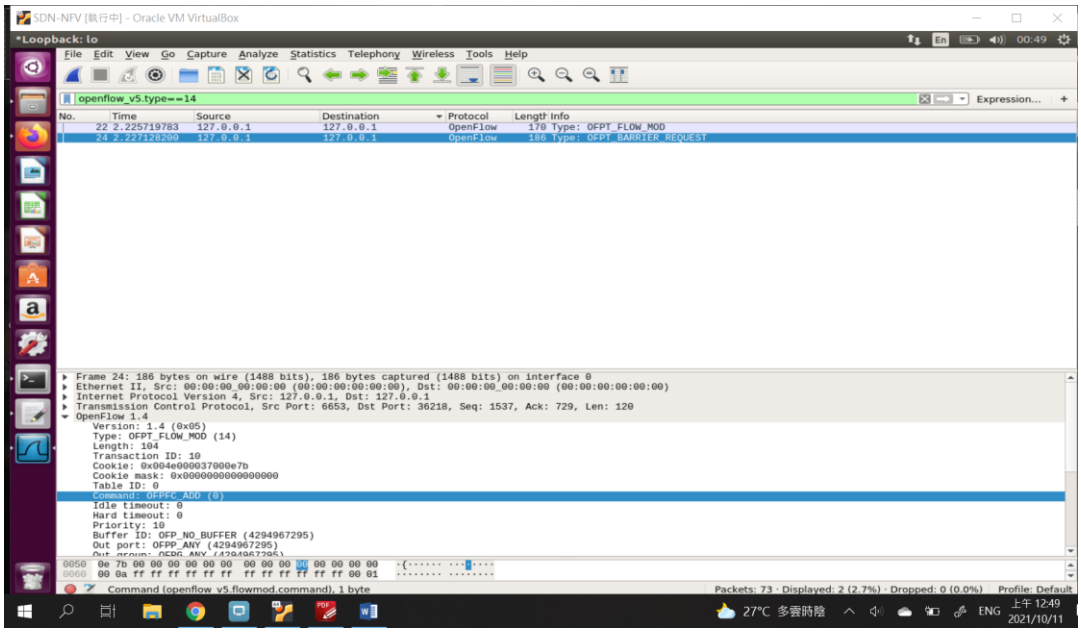


由上圖中可知，含有 OFPT_FLOW_MOD 的 packet 只有兩個，並且兩個 packet 的 command 皆為 OFPFC_ADD，如下圖

第一個封包 ：


SDN-NFV [執行中] - Oracle VM VirtualBox
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
openflow_v5.type==14
No. Time Source Destination Protocol Length Info
22 2.225719783 127.0.0.1 127.0.0.1 OpenFlow 170 Type: OFPT_FLOW_MOD
24 2.227128200 127.0.0.1 127.0.0.1 OpenFlow 186 Type: OFPT_BARRIER_REQUEST
Frame 22: 170 bytes on wire (1360 bits), 170 bytes captured (1360 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: 36218, Seq: 1433, Ack: 729, Len: 104
OpenFlow 1.4
Version: 1.4 (0x05)
Type: OFPT_FLOW_MOD (14)
Length: 104
Transaction ID: 9
Cookie: 0x004e00003dff173d
Cookie mask: 0x0000000000000000
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)
Command (openflow_v5.flowmod.command), 1 byte
Packets: 73 · Displayed: 2 (2.7%) · Dropped: 0 (0.0%)
Profile: Default
27°C 多雲時晴
ENG
上午 12:48
2021/10/11

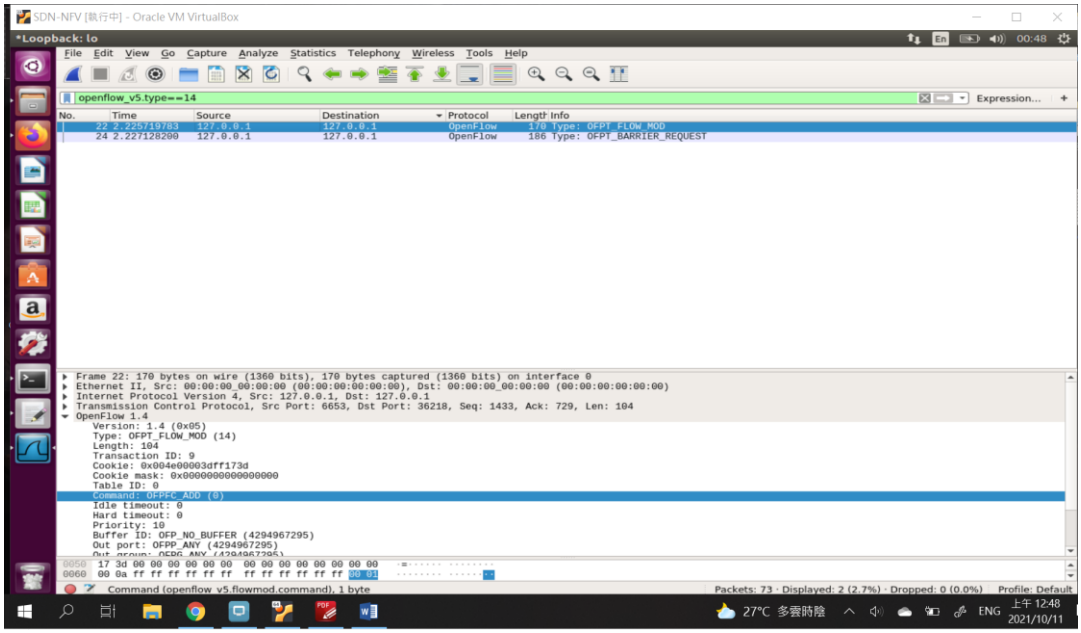
第二個封包


SDN-NFV [執行中] - Oracle VM VirtualBox
*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
openflow_v5.type==14
No. Time Source Destination Protocol Length Info
22 2.225719783 127.0.0.1 127.0.0.1 OpenFlow 170 Type: OFPT_FLOW_MOD
24 2.227128200 127.0.0.1 127.0.0.1 OpenFlow 186 Type: OFPT_BARRIER_REQUEST
Frame 24: 186 bytes on wire (1488 bits), 186 bytes captured (1488 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: 36218, Seq: 1537, Ack: 729, Len: 120
OpenFlow 1.4
Version: 1.4 (0x05)
Type: OFPT_FLOW_MOD (14)
Length: 104
Transaction ID: 10
Cookie: 0x004e000037000e7b
Cookie mask: 0x0000000000000000
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)
Command (openflow_v5.flowmod.command), 1 byte
Packets: 73 · Displayed: 2 (2.7%) · Dropped: 0 (0.0%)
Profile: Default
27°C 多雲時晴
ENG
上午 12:49
2021/10/11

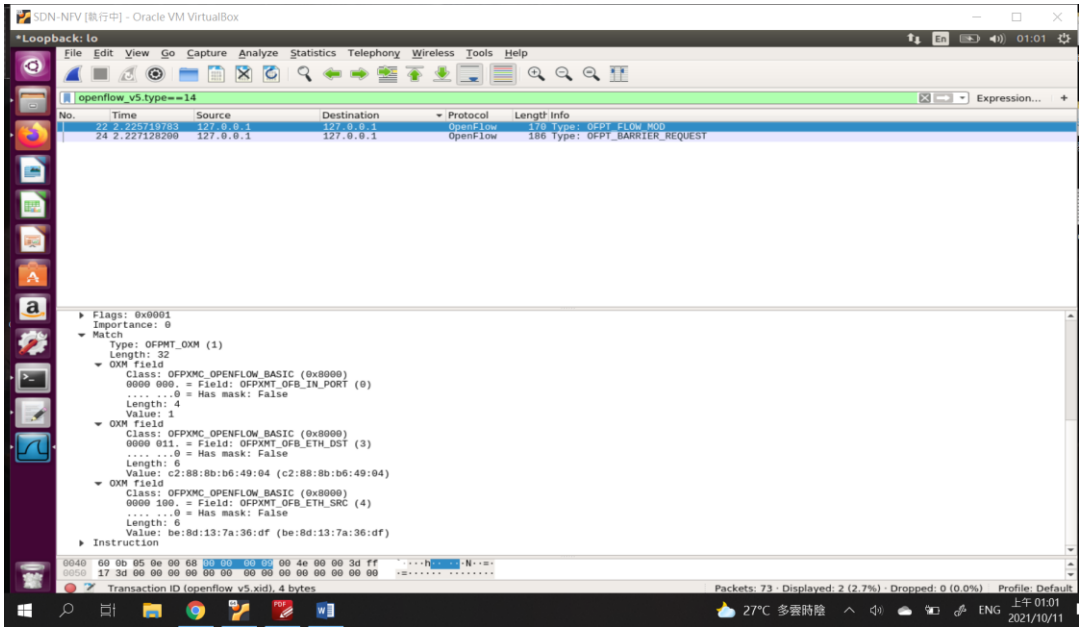
Q2:

根據 Q1，兩個封包的 match field 和 action 如下圖

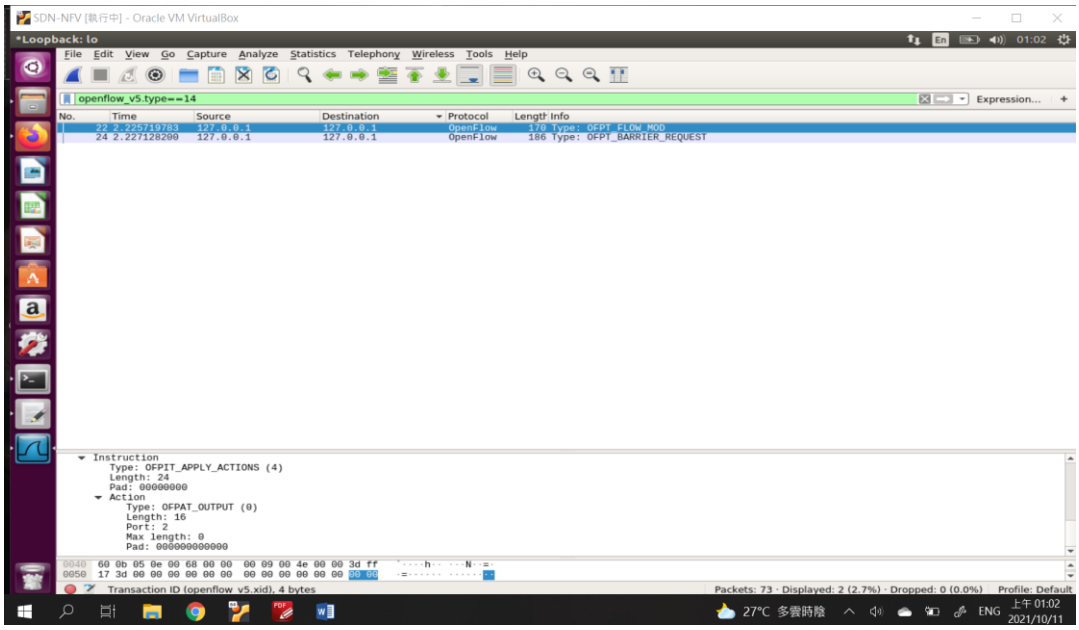
第一個封包 ：



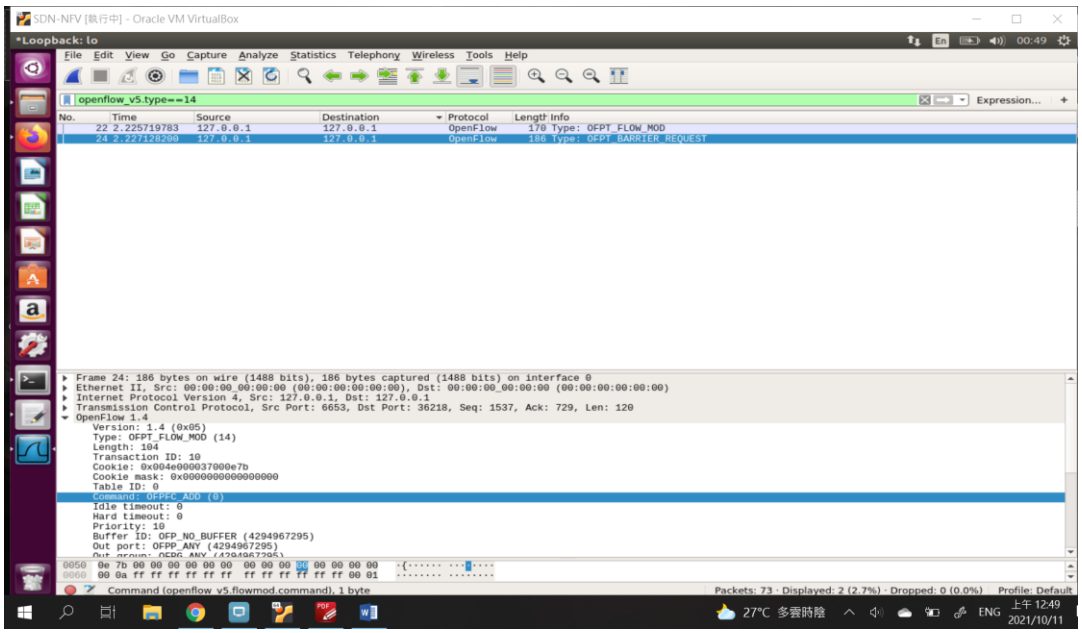
Match field :



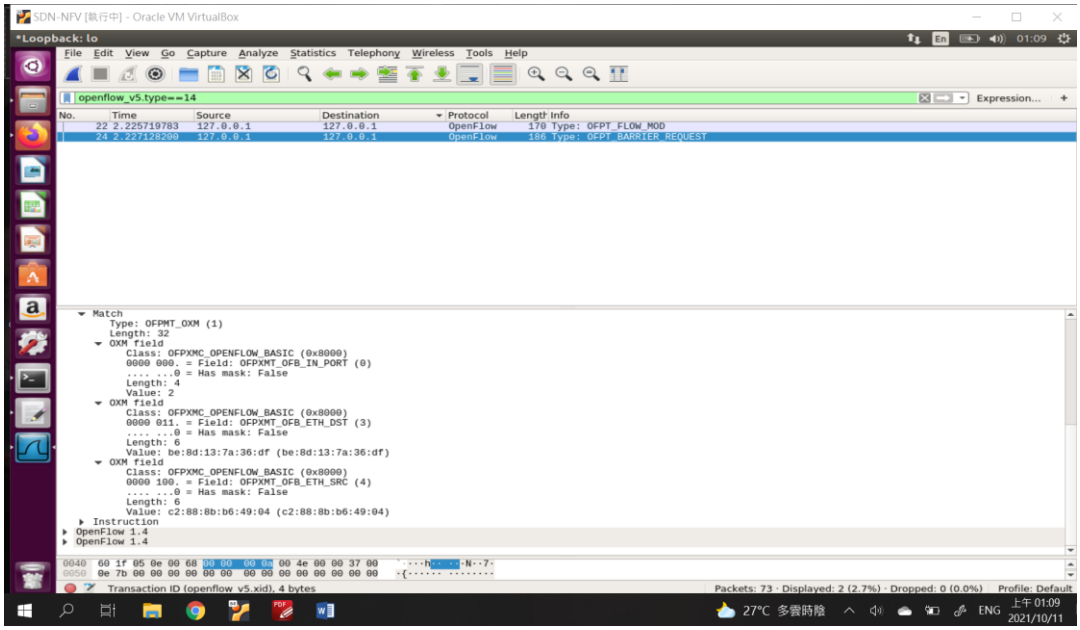
Action :



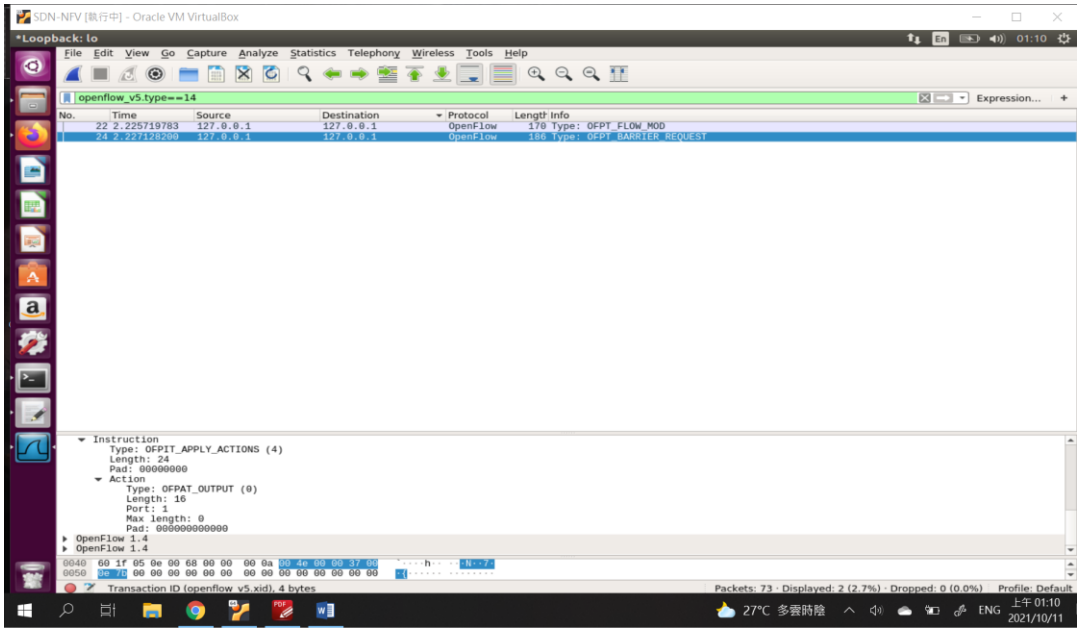
第二個封包 ：



Match :

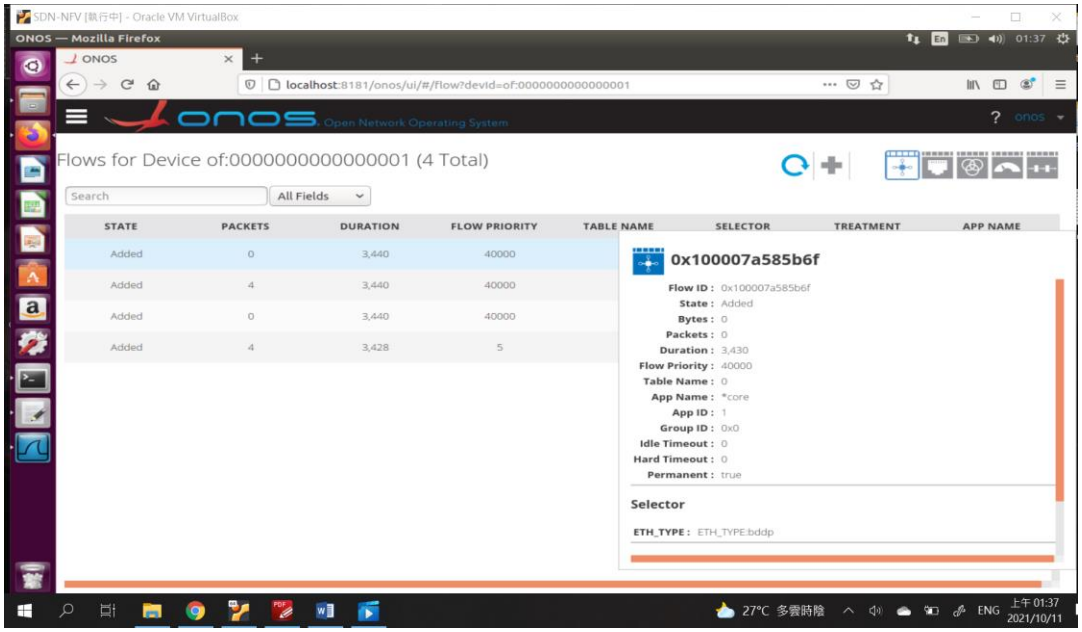


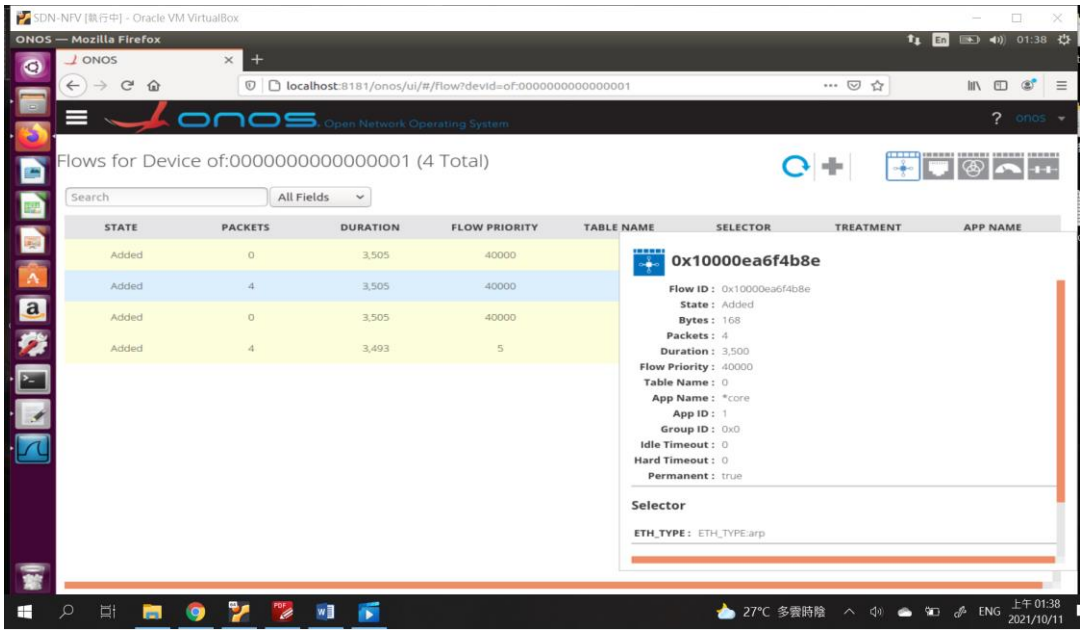
Action :

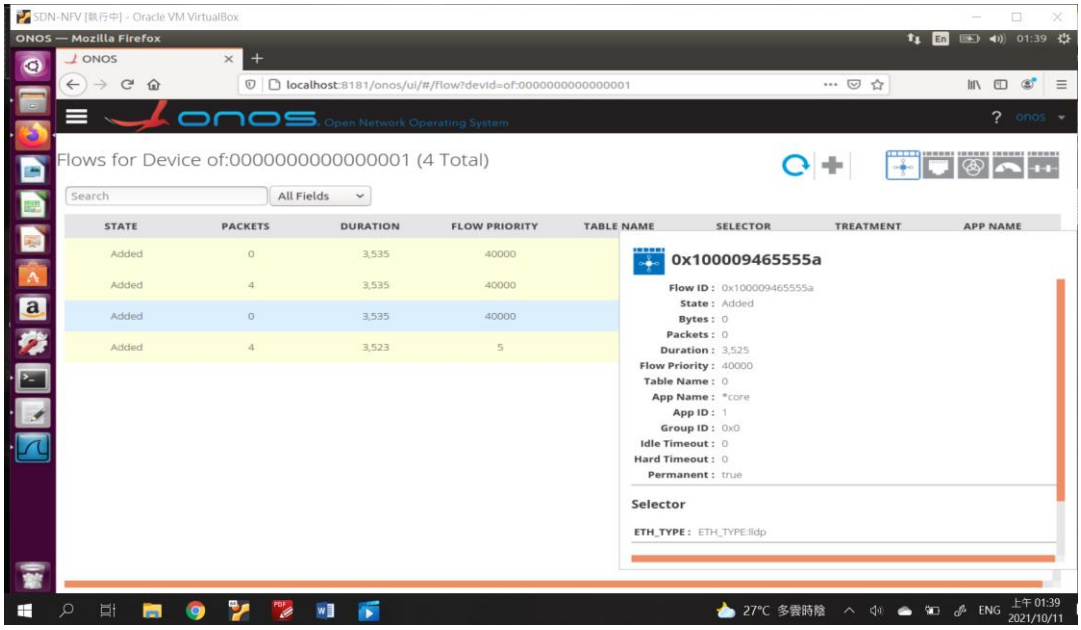


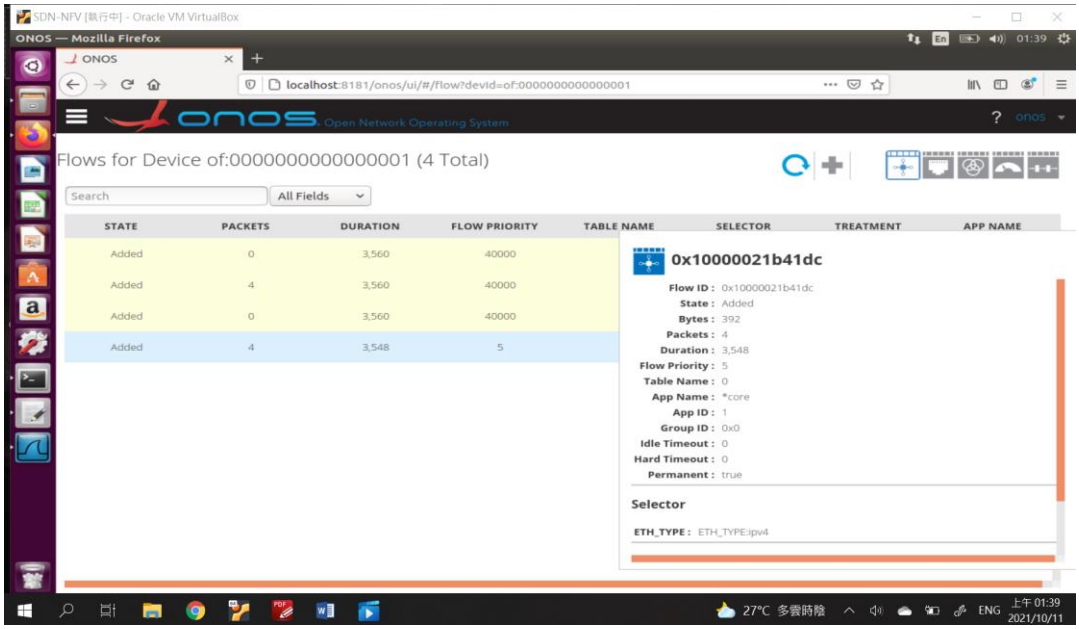
Q3 :

根據下圖，s1 中共有四條 rule，各自的 flow rule timeout 如下，可知他們各自的 timeout(idle timeout 和 hard timeout)皆為 0



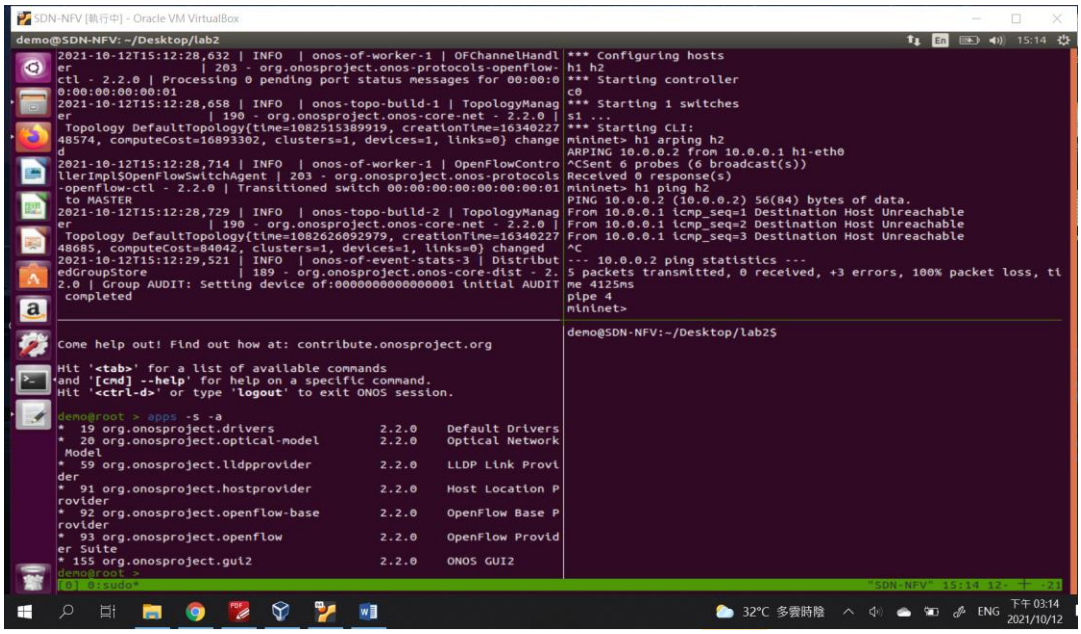



SDN-NFV [執行中] - Oracle VM VirtualBox
ONOS — Mozilla Firefox
ONOS
localhost:8181/onos/ui/#/flow?devId=of:0000000000000001
Open Network Operating System
Flows for Device of:0000000000000001 (4 Total)
Search
All Fields
STATE PACKETS DURATION FLOW PRIORITY TABLE NAME SELECTOR TREATMENT APP NAME
Added 0 3,535 40000
Added 4 3,535 40000
Added 0 3,535 40000
Added 4 3,523 5
0x100009465555a
Flow ID : 0x100009465555a
State : Added
Bytes : 0
Packets : 0
Duration : 3,525
Flow Priority : 40000
Table Name : 0
App Name : *core
App ID : 1
Group ID : 0x0
Idle Timeout : 0
Hard Timeout : 0
Permanent : true
Selector
ETH_TYPE : ETH_TYPE:lldp
27°C 多雲時晴
ENG
上午 01:39
2021/10/11


SDN-NFV [執行中] - Oracle VM VirtualBox
ONOS — Mozilla Firefox
ONOS
localhost:8181/onos/ui/#/flow?devId=of:0000000000000001
Open Network Operating System
Flows for Device of:0000000000000001 (4 Total)
Search
All Fields
STATE PACKETS DURATION FLOW PRIORITY TABLE NAME SELECTOR TREATMENT APP NAME
Added 0 3,560 40000
Added 4 3,560 40000
Added 0 3,560 40000
Added 4 3,548 5
0x10000021b41dc
Flow ID : 0x10000021b41dc
State : Added
Bytes : 392
Packets : 4
Duration : 3,548
Flow Priority : 5
Table Name : 0
App Name : *core
App ID : 1
Group ID : 0x0
Idle Timeout : 0
Hard Timeout : 0
Permanent : true
Selector
ETH_TYPE : ETH_TYPE:ipv4
27°C 多雲時晴
ENG
上午 01:39
2021/10/11

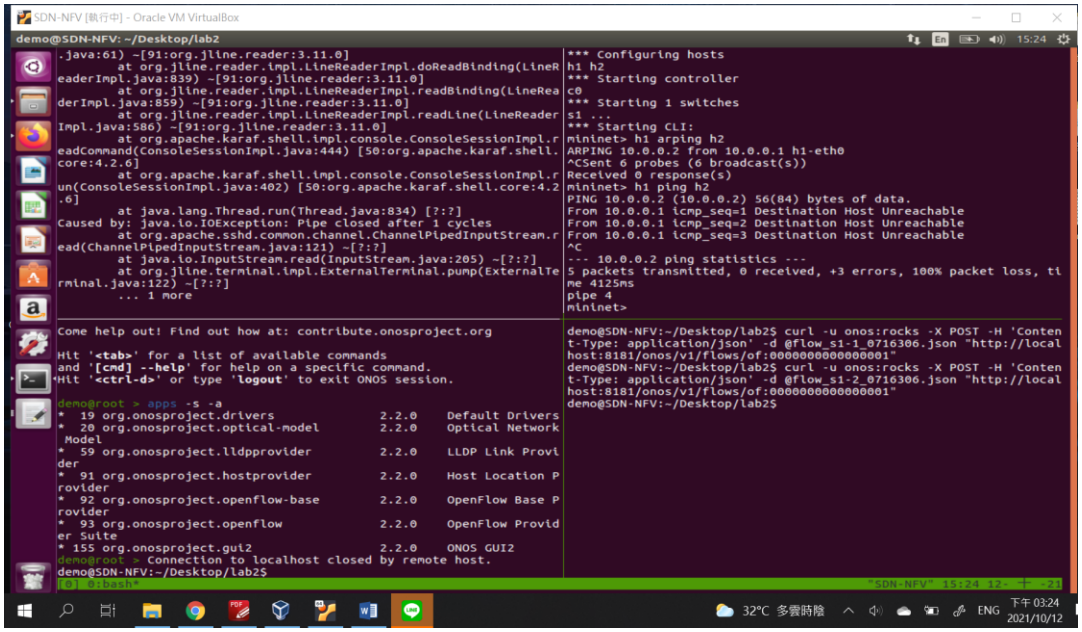
Part 2

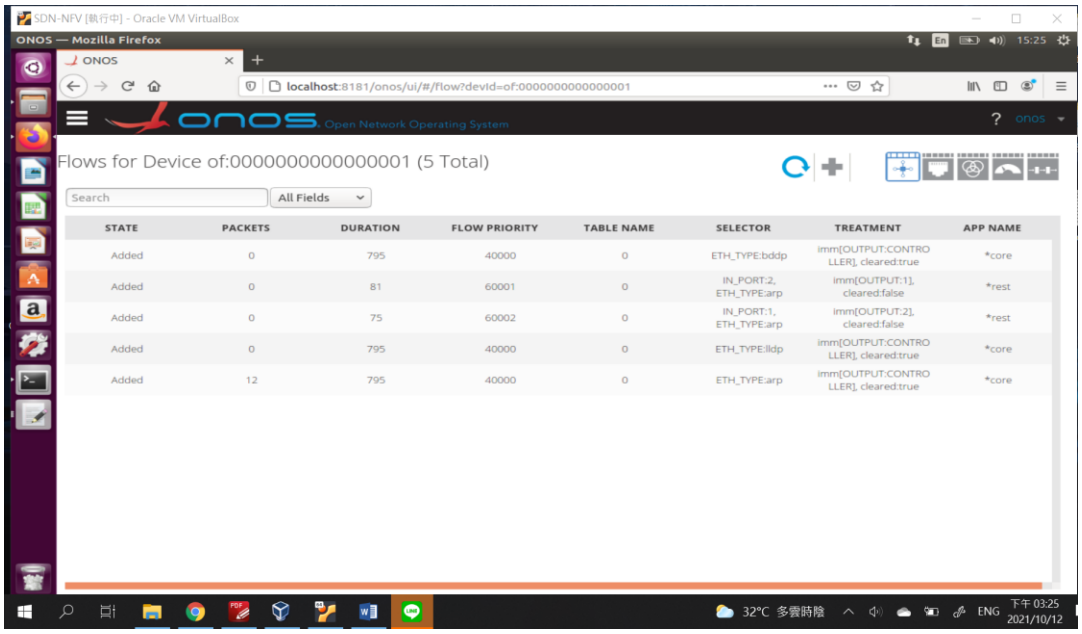
在完成 part2 的準備工作後，測試 arping 和 ping 的結果，如下圖，可知目前 ARP packet 和 IP packet 皆無法在 h1 和 h2 之間流通



首先處理 ARP 的部分，將 flow1.json 改寫成下列兩個 json file(為容易在 onos 上辨別 rule，每個 rule 各自採用不同的 priority)，左方的 file 是讓 h2 到 h1 的 ARP packet 可以通行，而右方的 file 作用於相反的方向，讓 h1 的 ARP packet 可以到達 h2。因為 spec 指定 match field 需要有 ARP 的 ethernet type，因此新增了 ethernet type 為 ARP(0X0806)的 criteria

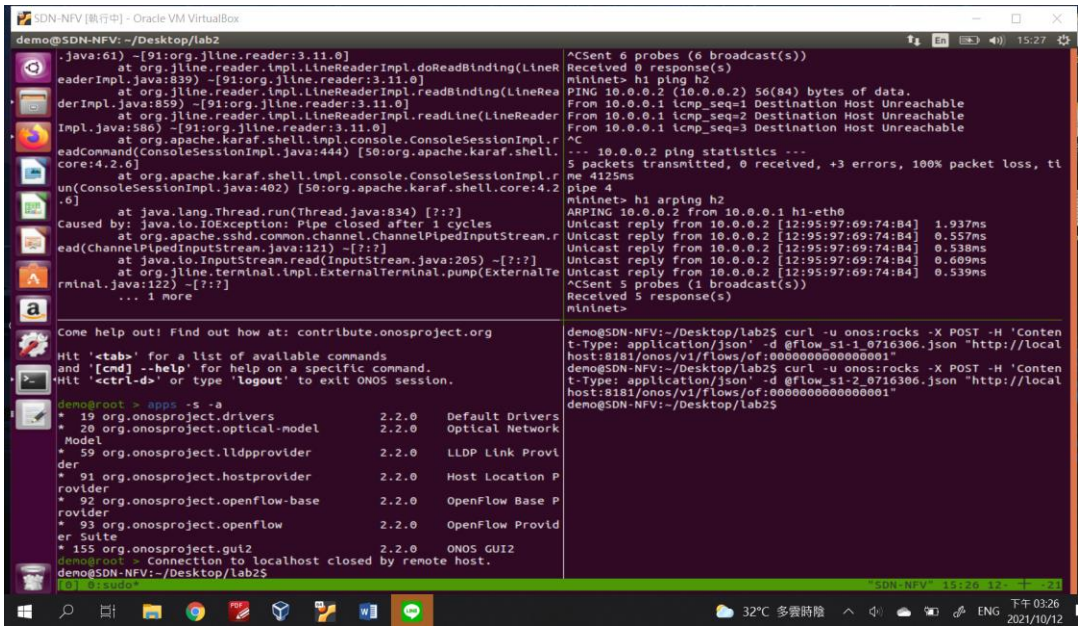
將兩個 json file 用 curl 傳輸到 ONOS，可看見在 ONOS 上已經有我們指定的 flow(priority 為 60001 和 60002)



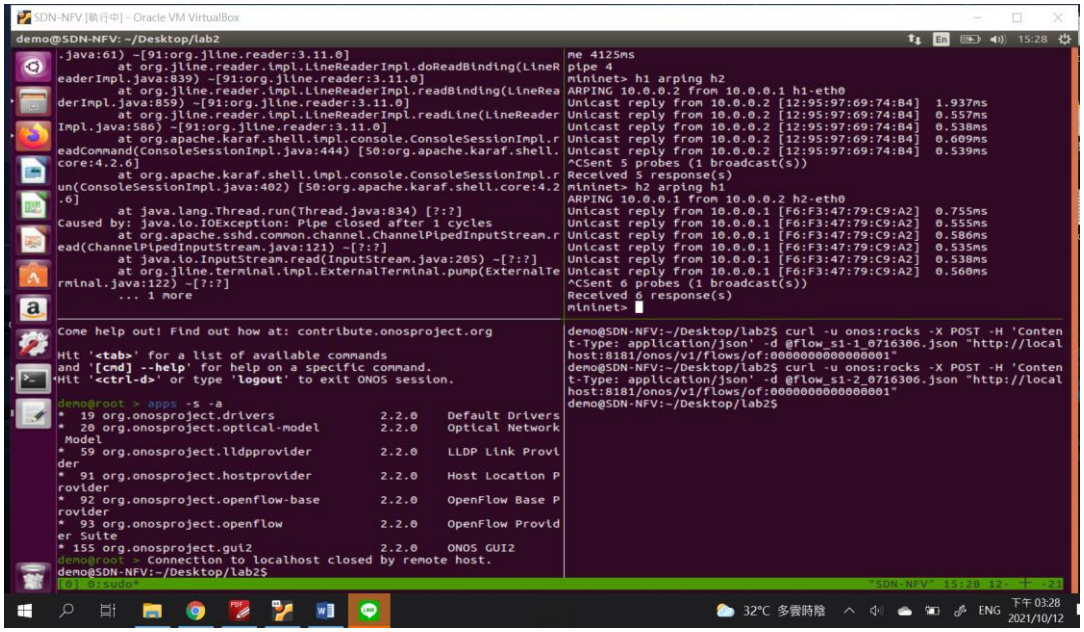


嘗試使用 h1 arping h2 和 h2 arping h1 得結果如下，可知 h1 和 h2 皆可成功使用 arping 找到對方

h1 arping h2:

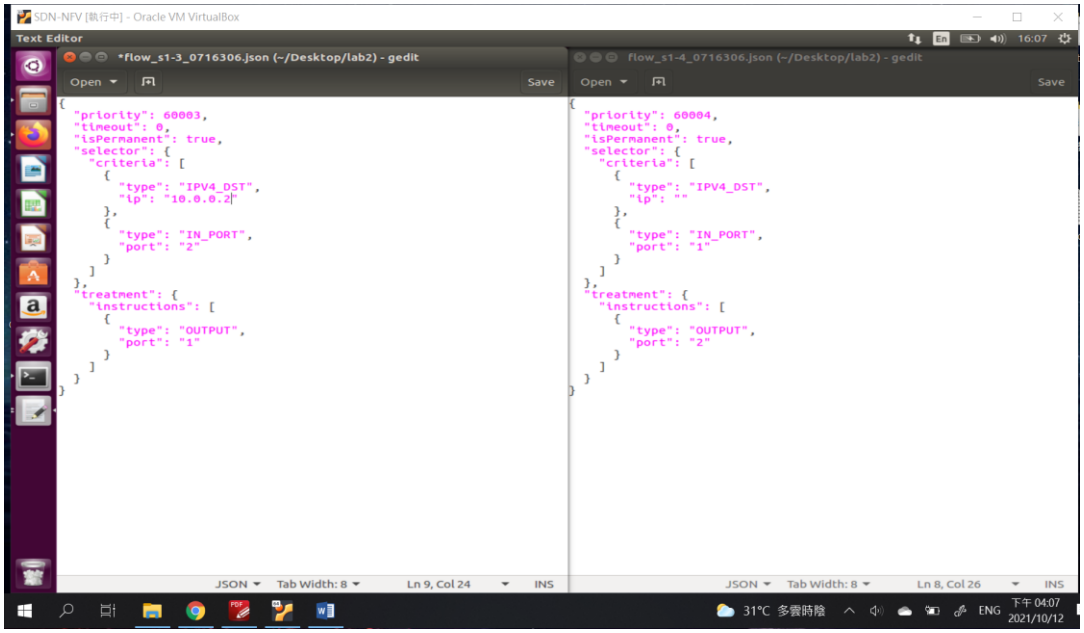


h2 arping h1:

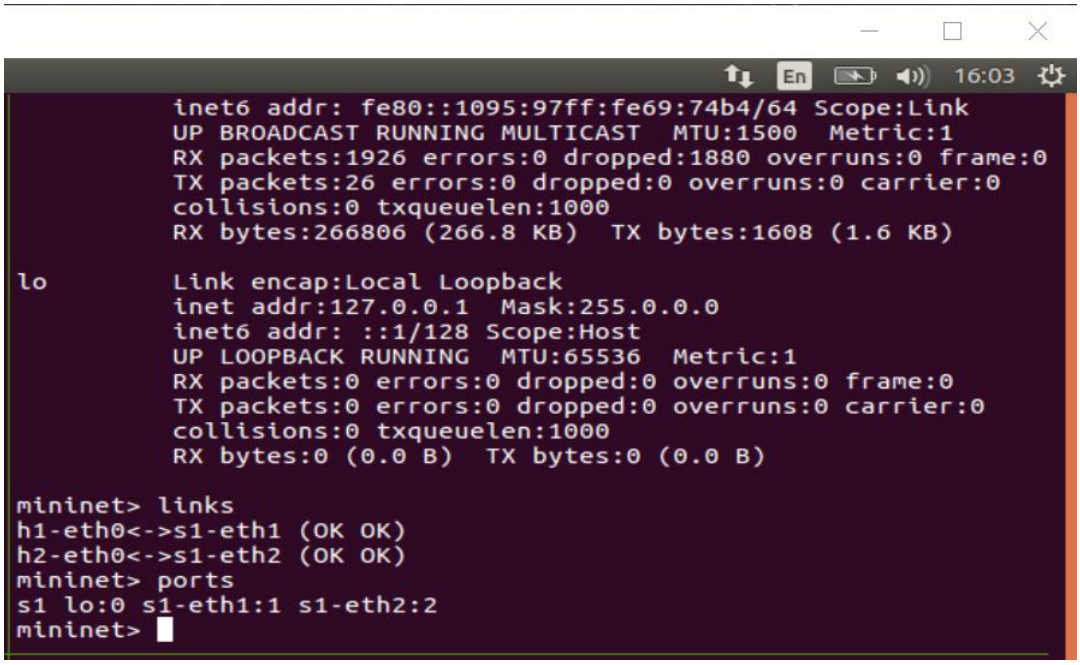


接下來完成 IPv4 packet 的部分，下圖為兩個 json file

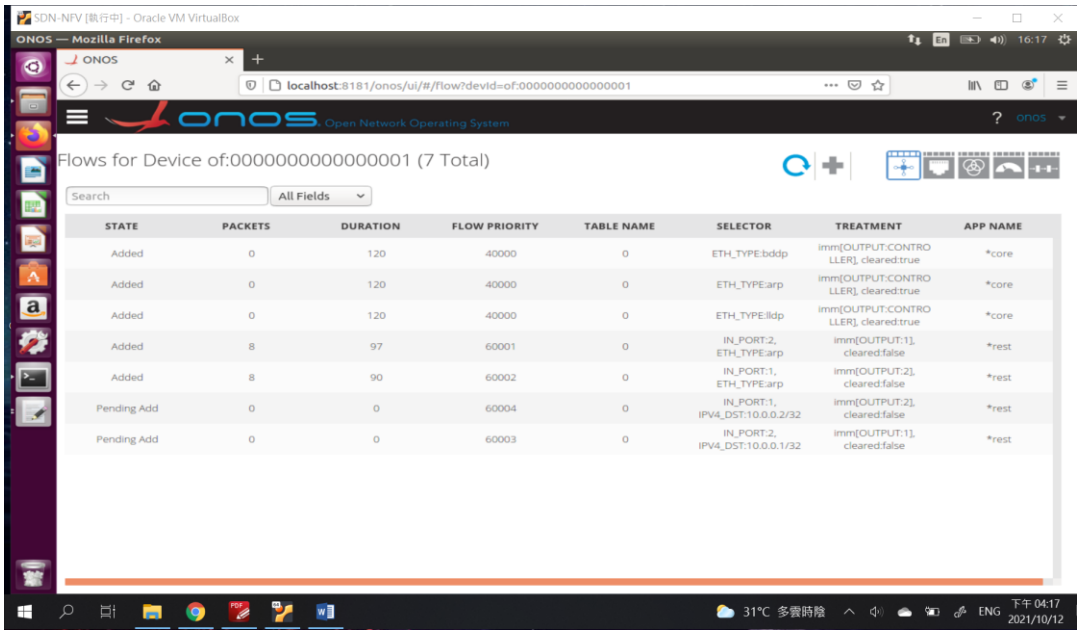
和前兩個負責 ARP packet 的 file 的不同處在於，flow 的篩選標準需要從 ethernet type 改成 IPv4 destination，因此 match field 的 ethType 需要改成 IPV4_DST，並各自根據進入 switch 的 port 和 IPv4 destination 判斷 flow 的方向，並且使用 priority 為 60003 和 60004 以和前兩個 flow rule 區別



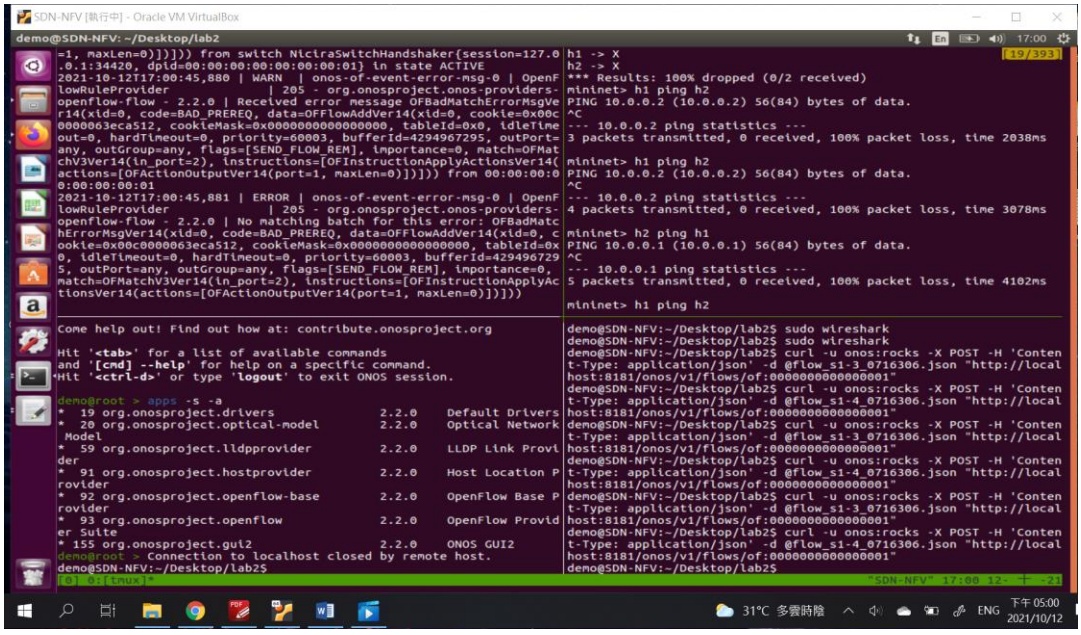
由下圖中可知，h1 連接到 s1-eth1 且 h2 連接到 s1-eth2，而 s1-eth1 使用的 port 為 1、s1-eth2 使用的 port 為 2，因此上圖中左方的 file 是 h2 到 h1 的 flow、右方的 file 是 h1 到 h2 的 flow

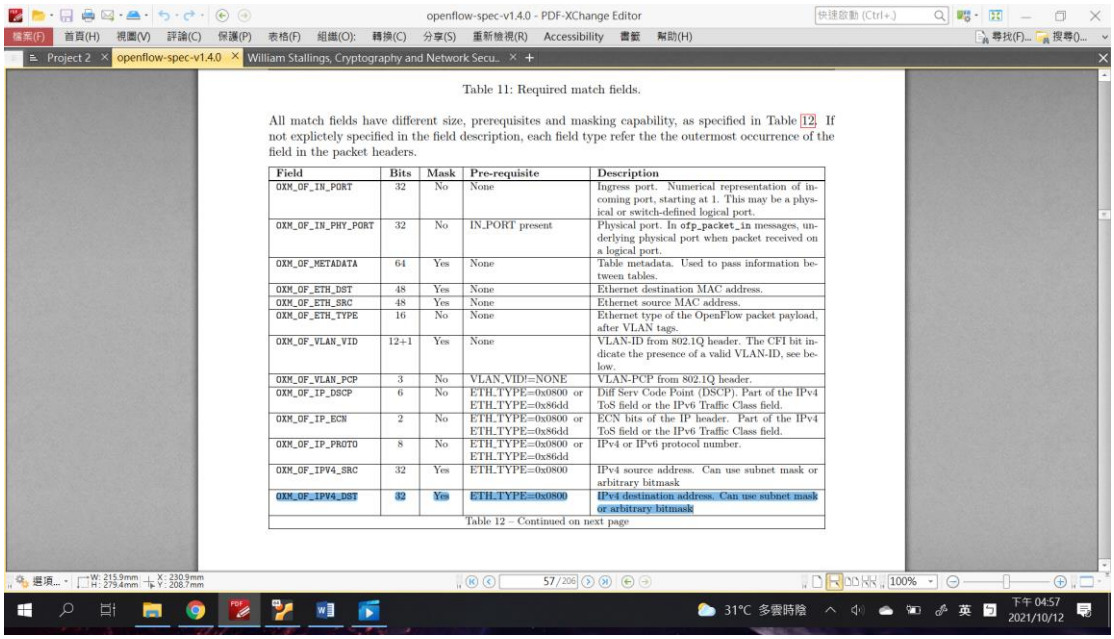


一樣用 curl 將兩個 file 傳送到 ONOS 後，可看見 ONOS 上已有 flow rule 存在

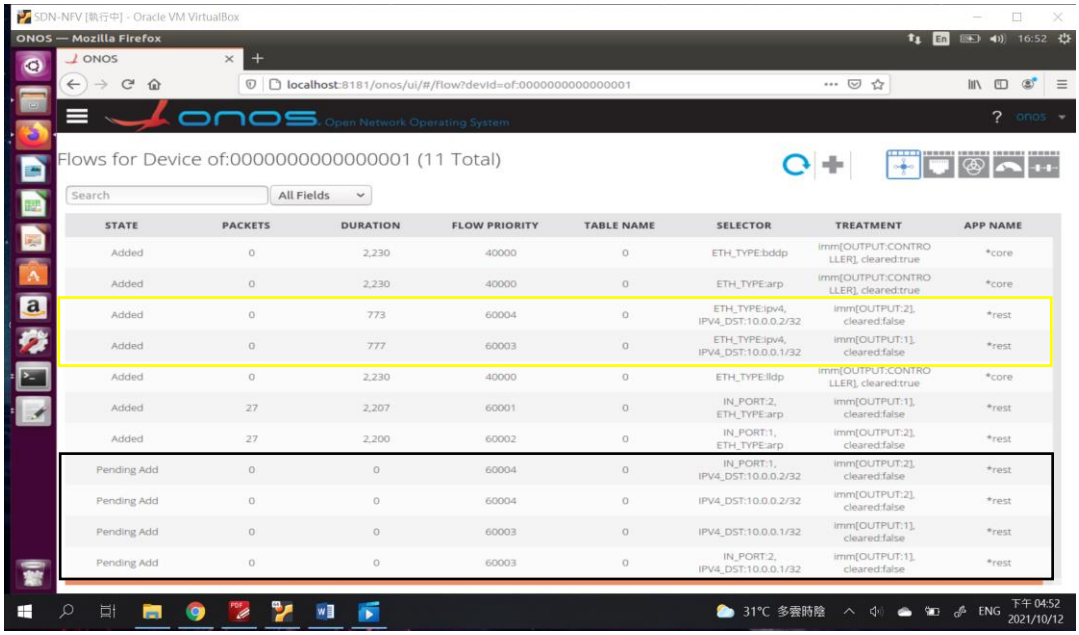


但之後無論是使用 h1 ping h2 或 h2 ping h1 都無法成功，根據 openflow spec 中的描述，原因是因為 IPV4_DST 有 dependency，需要同時有 ETH_TYPE=”0x0800” 這個 criteria 存在

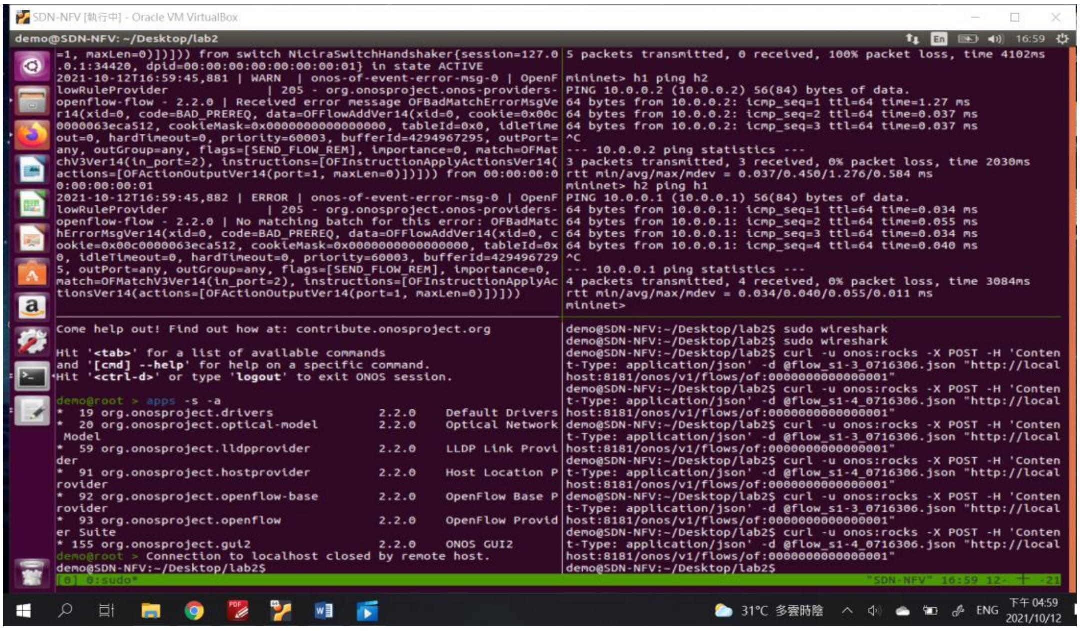




因此在下圖中，可看見黑色外框中的 flow rule 因為沒有加上 ETH_TYPE 作為 criteria，因此不會有 packet 使用(持續在 pending add 狀態)，而黃色外框中的 rule 因為有加上 ETH_TYPE，因此會有 flow 經過(從 pending add 轉換為 added 的狀態)

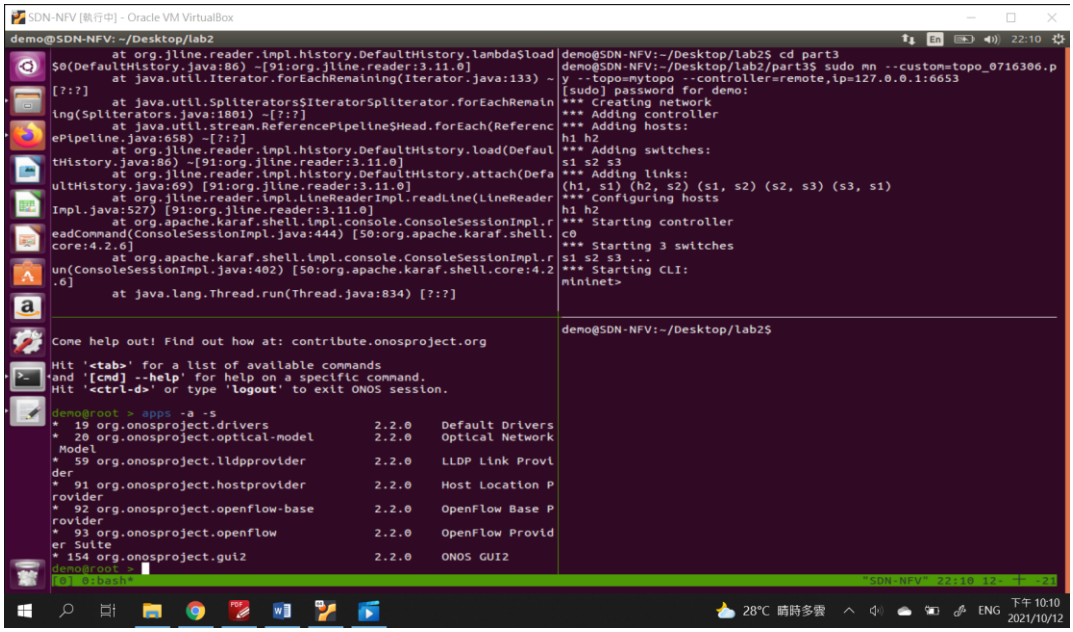


放入 ETH_TYPE 作為 criteria 後，可看見 h1 ping h2 和 h2 ping h1 都成功



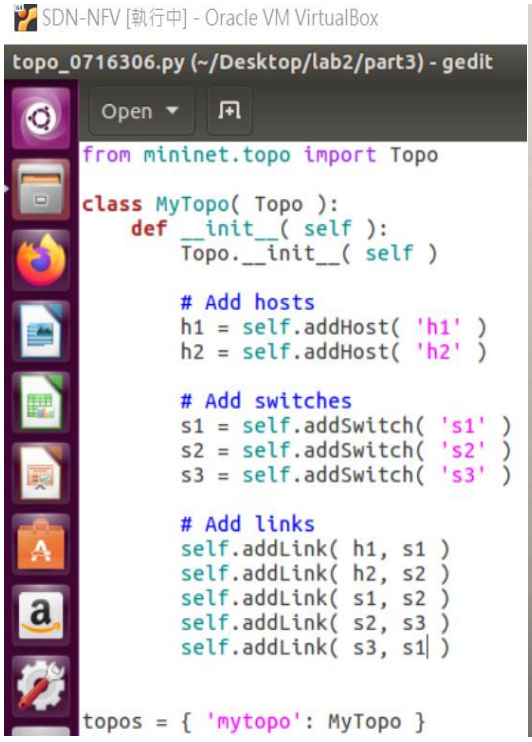
Part3

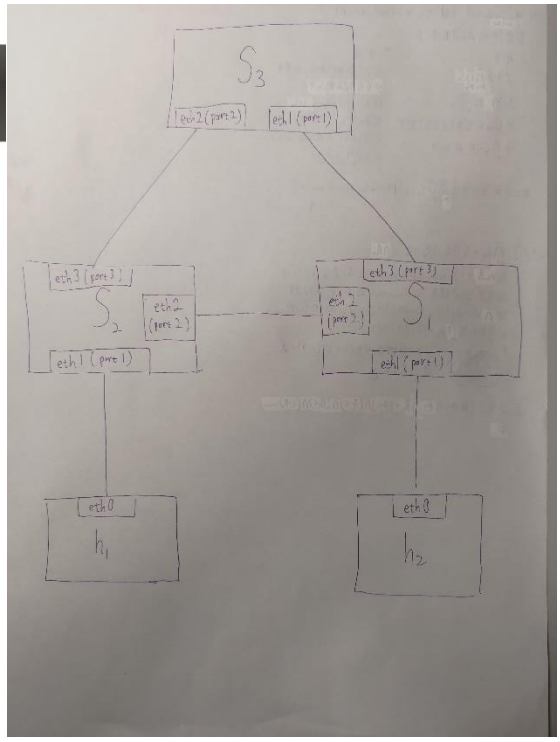
開啟 ONOS 後，確認無其他多餘的 app 被開啟



因為 broadcast storm 是起因於 ARP packet 的 broadcast 導致，若多個 switch 相連並且形成迴圈(loop)，則會發生 broadcast storm

以下是我設計的 topology，為方便設計，只使用 3 個 switch 形成一個 loop，code 及示意圖如下

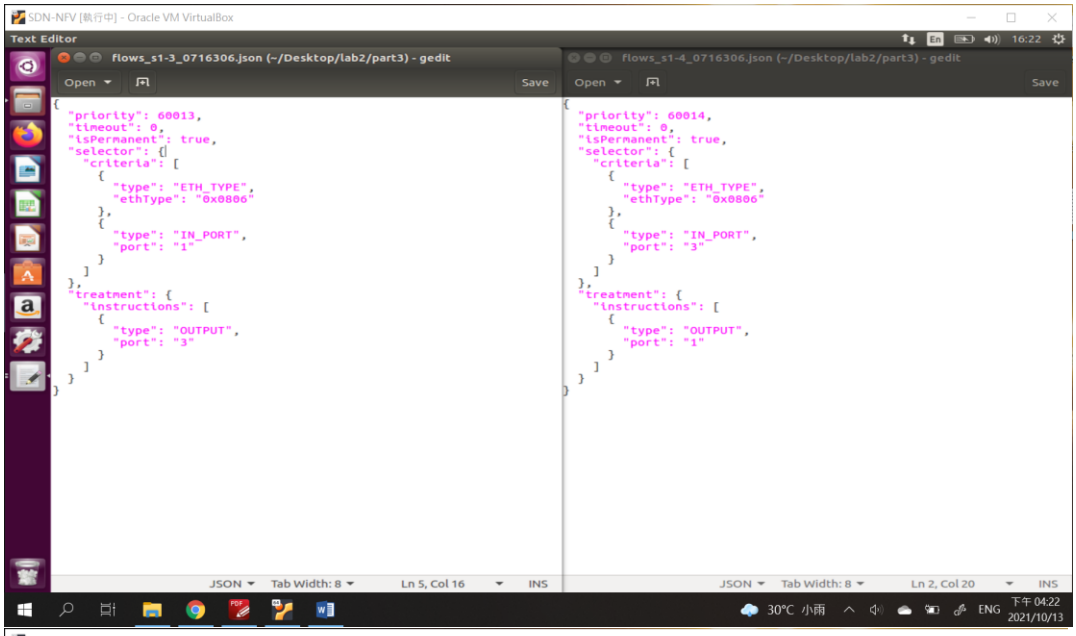


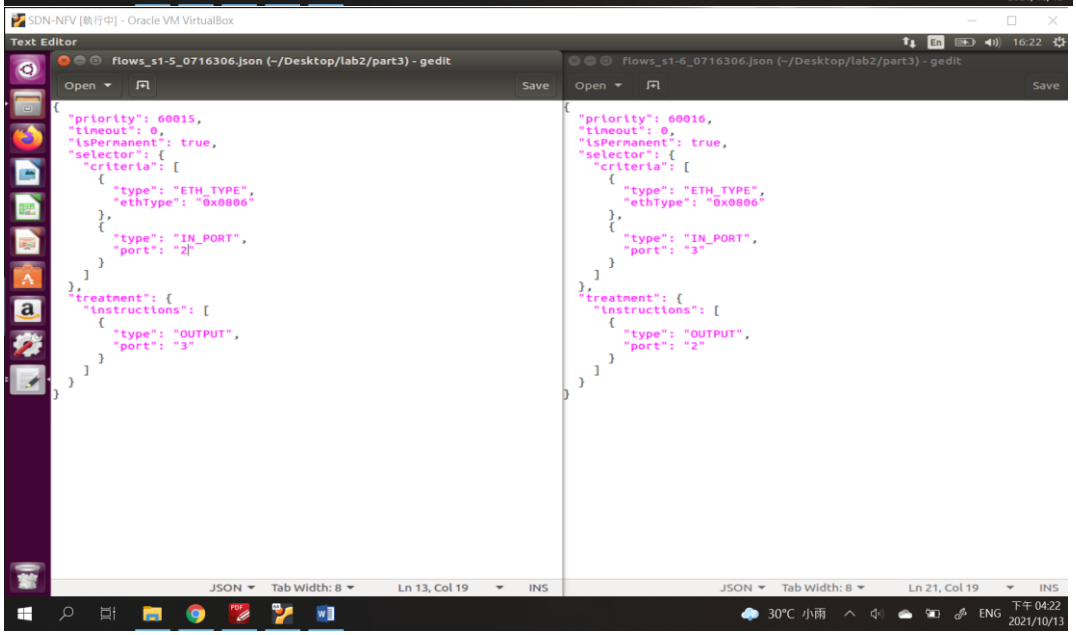


```
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (s1, s2) (s2, s3) (s3, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s2-eth1 (OK OK)
s1-eth2<->s2-eth2 (OK OK)
s2-eth3<->s3-eth1 (OK OK)
s3-eth2<->s1-eth3 (OK OK)
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3
s2 lo:0 s2-eth1:1 s2-eth2:2 s2-eth3:3
s3 lo:0 s3-eth1:1 s3-eth2:2
mininet>
```

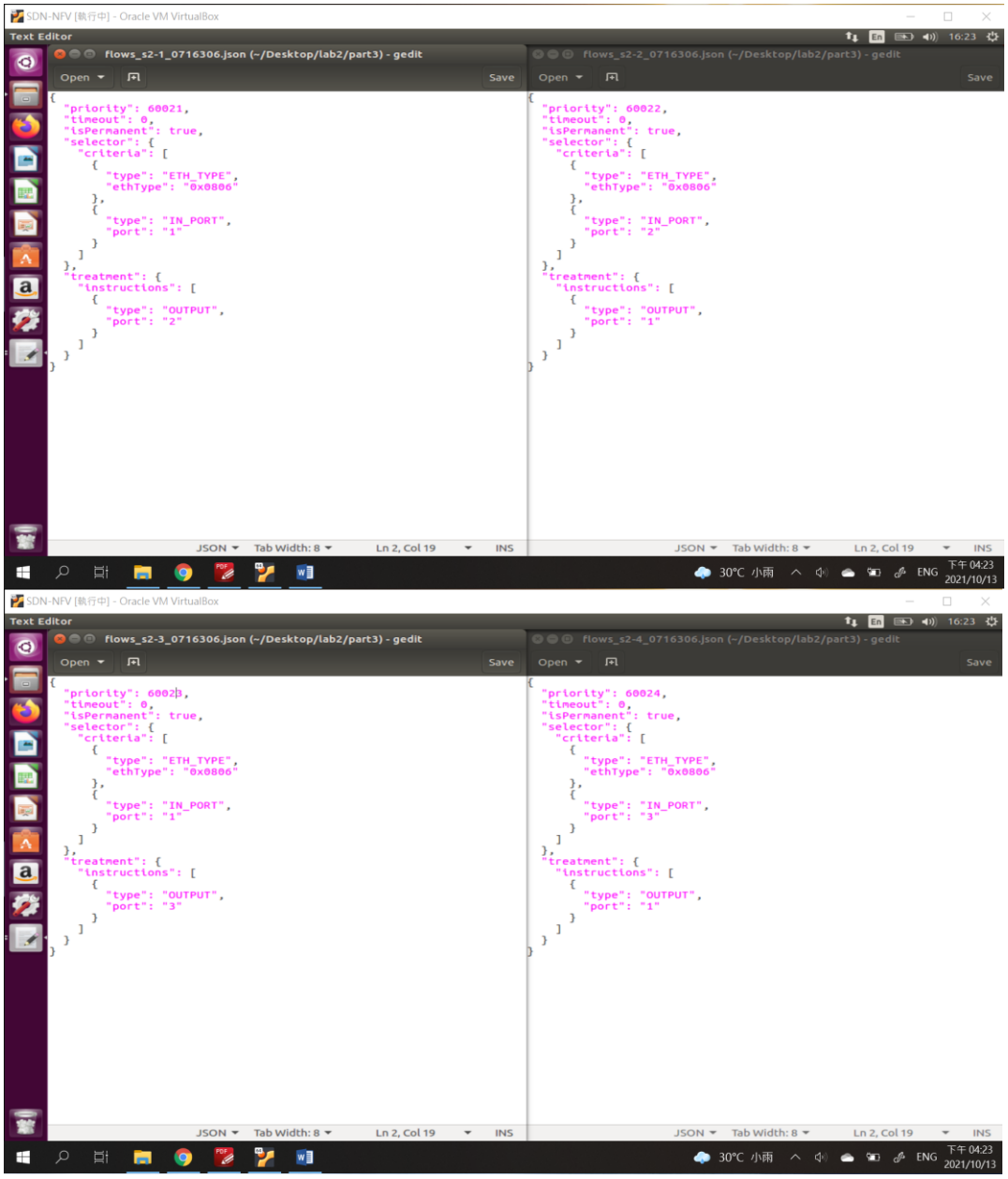
使用的 flow rule 如下，因為每個方向的 flow 都要一個 json file，故在 switch 中 s1 和 s2 因為額外連接了 h1 和 h2，各自有 6 個 rule，分別放在 flows_s1-x_0716306.json 和 flows_s2-x_0716306.json(其中 x 為 1～6)，而 s3 只連接 s1 和 s2，因此只需要 2 個 rule，分別為 flows_s3-1_0716306.json 和 flows_s3-2_0716306.json，使得一個 switch 若有 ARP packet 進入，能將這個 packet 送到另一個 switch，並且能讓 h1 arping h2 或 h2 arping h1 成功

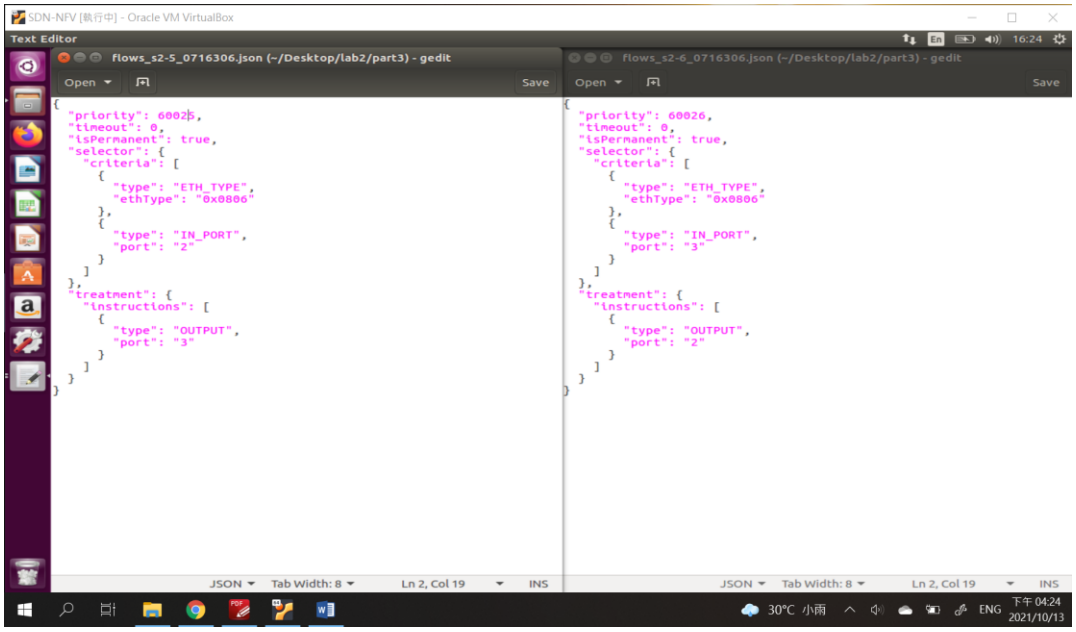
s1 的 rule


SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s1-3_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60013,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "1"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "3"
}
]
}
}
JSON
Tab Width: 8
Ln 5, Col 16
INS
flows_s1-4_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60014,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "3"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "1"
}
]
}
}
JSON
Tab Width: 8
Ln 2, Col 20
INS
30°C 小雨
ENG
下午 04:22
2021/10/13

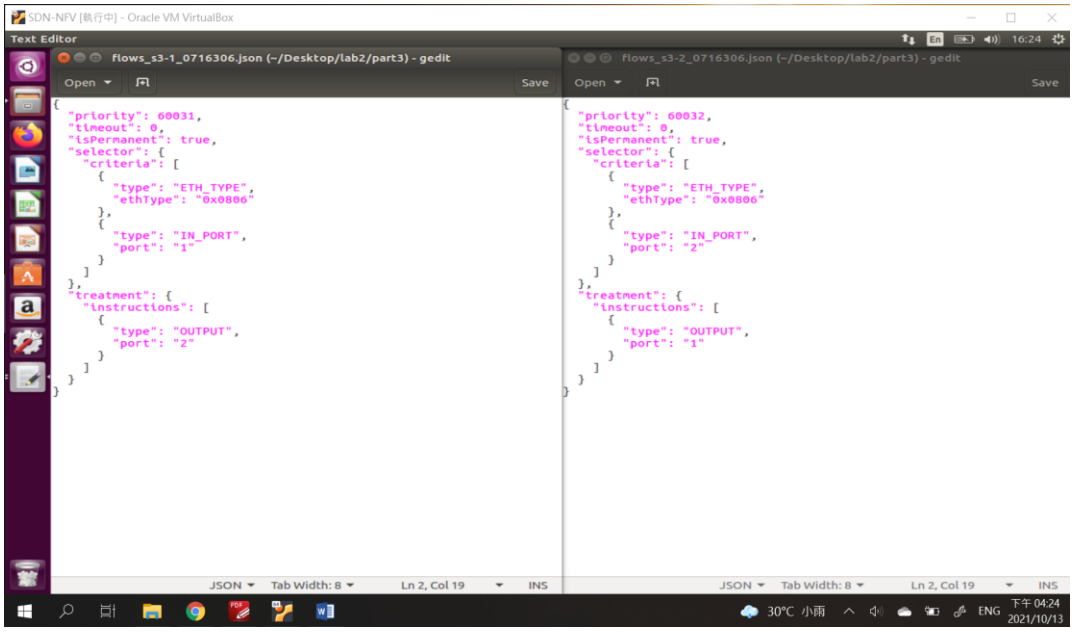

SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s1-5_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60015,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "2"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "3"
}
]
}
}
JSON
Tab Width: 8
Ln 13, Col 19
INS
flows_s1-6_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60016,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "3"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "2"
}
]
}
}
JSON
Tab Width: 8
Ln 21, Col 19
INS
30°C 小雨
ENG
下午 04:22
2021/10/13

s2 的 rule


SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s2-1_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60021,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "1"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "2"
}
]
}
}
flows_s2-2_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60022,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "2"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "1"
}
]
}
}
JSON
Tab Width: 8
Ln 2, Col 19
INS
30°C 小雨
ENG
下午 04:23
2021/10/13
SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s2-3_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60023,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "1"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "3"
}
]
}
}
flows_s2-4_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60024,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "3"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "1"
}
]
}
}
JSON
Tab Width: 8
Ln 2, Col 19
INS
30°C 小雨
ENG
下午 04:23
2021/10/13


SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s2-5_0716306.json (~/Desktop/lab2/part3) - gedit
flows_s2-6_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60025,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "2"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "3"
}
]
}
}
{
"priority": 60026,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "3"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "2"
}
]
}
}
JSON
Tab Width: 8
Ln 2, Col 19
INS
30°C 小雨
ENG
下午 04:24
2021/10/13

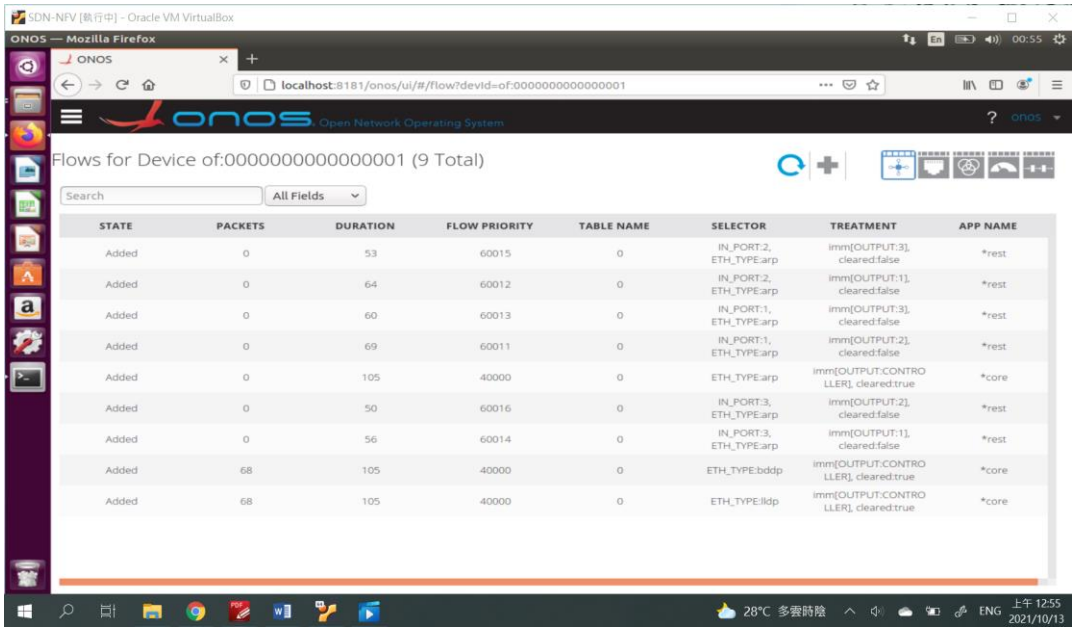
s3 的 rule


SDN-NFV [執行中] - Oracle VM VirtualBox
Text Editor
flows_s3-1_0716306.json (~/Desktop/lab2/part3) - gedit
flows_s3-2_0716306.json (~/Desktop/lab2/part3) - gedit
Open
Save
{
"priority": 60031,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "1"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "2"
}
]
}
}
{
"priority": 60032,
"timeout": 0,
"isPermanent": true,
"selector": {
"criteria": [
{
"type": "ETH_TYPE",
"ethType": "0x0806"
},
{
"type": "IN_PORT",
"port": "2"
}
]
},
"treatment": {
"instructions": [
{
"type": "OUTPUT",
"port": "1"
}
]
}
}
JSON
Tab Width: 8
Ln 2, Col 19
INS
30°C 小雨
ENG
下午 04:24
2021/10/13

使用 curl 將每個 switch 的 flow 傳送到 ONOS 上

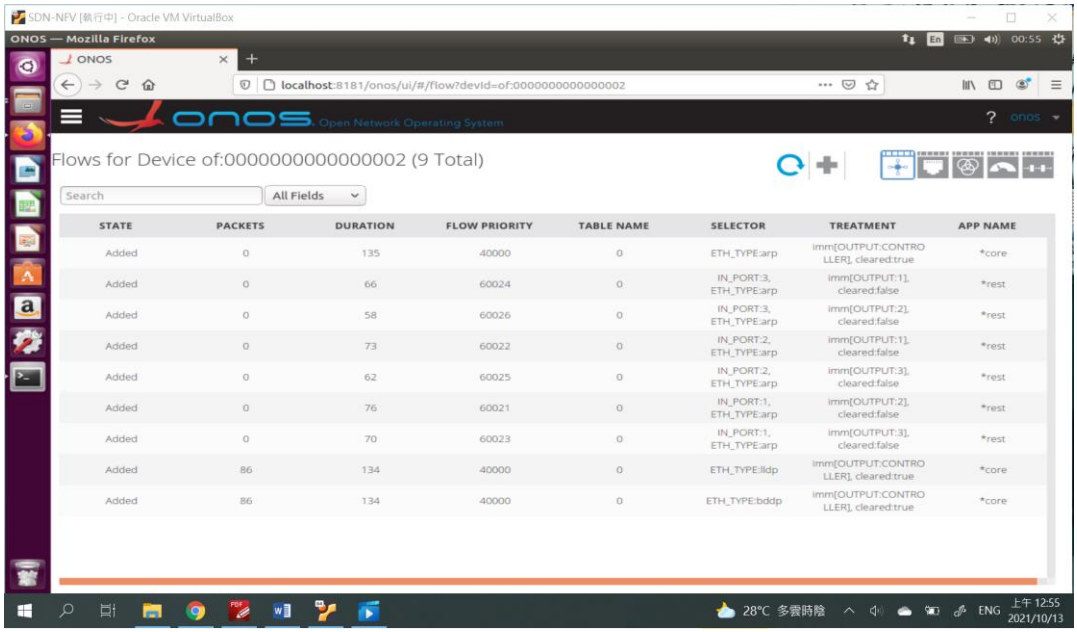
s1:

```
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-1_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-2_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-3_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-4_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-5_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s1-6_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000001'
demo@SDN-NFV:~/Desktop/lab2/part3$
                                                    "SDN-NFV" 00:34 13- -21
```

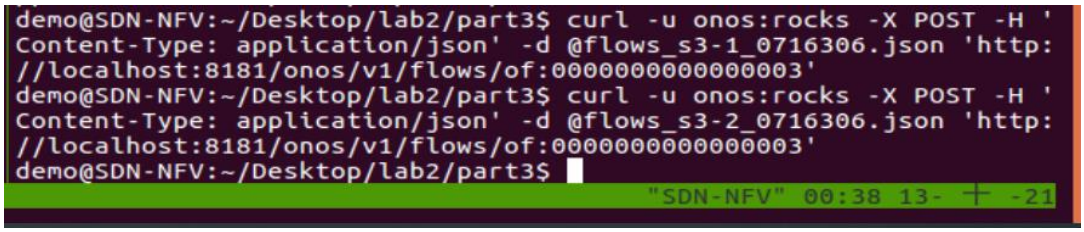


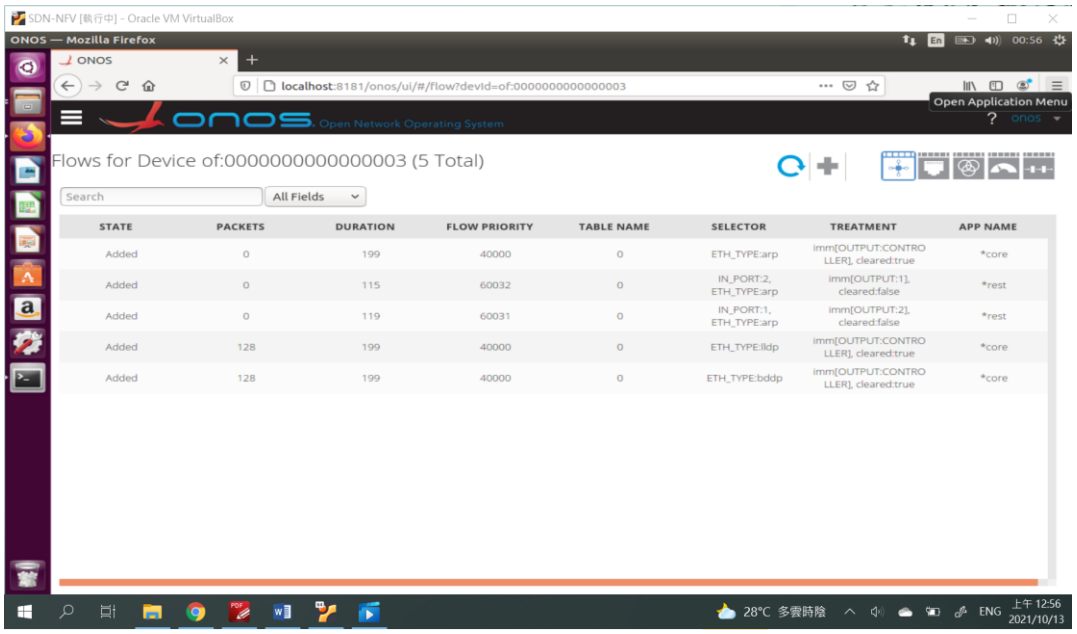
s2

```
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-1_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-2_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-3_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-4_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-5_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s2-6_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000002'
demo@SDN-NFV:~/Desktop/lab2/part3$
                                                    "SDN-NFV" 00:36 13- -21
```


SDN-NFV [執行中] - Oracle VM VirtualBox
ONOS — Mozilla Firefox
ONOS
localhost:8181/onos/ui/#/flow?devId=of:0000000000000002
onos Open Network Operating System
Flows for Device of:0000000000000002 (9 Total)
Search
All Fields
STATE PACKETS DURATION FLOW PRIORITY TABLE NAME SELECTOR TREATMENT APP NAME
Added 0 135 40000 0 ETH_TYPE:arp imm[OUTPUT:CONTROLLER], cleared:true *core
Added 0 66 60024 0 IN_PORT:3, ETH_TYPE:arp imm[OUTPUT:1], cleared:false *rest
Added 0 58 60026 0 IN_PORT:3, ETH_TYPE:arp imm[OUTPUT:2], cleared:false *rest
Added 0 73 60022 0 IN_PORT:2, ETH_TYPE:arp imm[OUTPUT:1], cleared:false *rest
Added 0 62 60025 0 IN_PORT:2, ETH_TYPE:arp imm[OUTPUT:3], cleared:false *rest
Added 0 76 60021 0 IN_PORT:1, ETH_TYPE:arp imm[OUTPUT:2], cleared:false *rest
Added 0 70 60023 0 IN_PORT:1, ETH_TYPE:arp imm[OUTPUT:3], cleared:false *rest
Added 86 134 40000 0 ETH_TYPE:lldp imm[OUTPUT:CONTROLLER], cleared:true *core
Added 86 134 40000 0 ETH_TYPE:bddp imm[OUTPUT:CONTROLLER], cleared:true *core
28°C 多雲時晴
ENG 上午 12:55 2021/10/13

s3

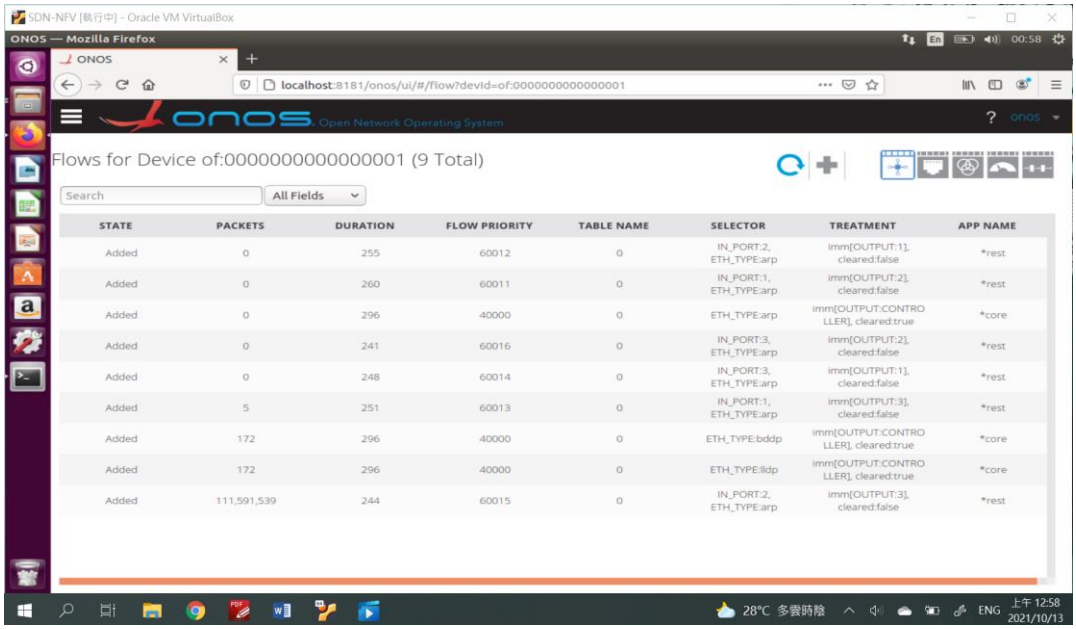

demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s3-1_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000003'
demo@SDN-NFV:~/Desktop/lab2/part3$ curl -u onos:rocks -X POST -H '
Content-Type: application/json' -d @flows_s3-2_0716306.json 'http:
//localhost:8181/onos/v1/flows/of:0000000000000003'
demo@SDN-NFV:~/Desktop/lab2/part3$
"SDN-NFV" 00:38 13-


SDN-NFV [執行中] - Oracle VM VirtualBox
ONOS — Mozilla Firefox
ONOS
localhost:8181/onos/ui/#/flow?devId=of:0000000000000003
Open Application Menu
onos Open Network Operating System
Flows for Device of:0000000000000003 (5 Total)
Search
All Fields
STATE PACKETS DURATION FLOW PRIORITY TABLE NAME SELECTOR TREATMENT APP NAME
Added 0 199 40000 0 ETH_TYPE:arp imm[OUTPUT:CONTROLLER], cleared:true *core
Added 0 115 60032 0 IN_PORT:2, ETH_TYPE:arp imm[OUTPUT:1], cleared:false *rest
Added 0 119 60031 0 IN_PORT:1, ETH_TYPE:arp imm[OUTPUT:2], cleared:false *rest
Added 128 199 40000 0 ETH_TYPE:lldp imm[OUTPUT:CONTROLLER], cleared:true *core
Added 128 199 40000 0 ETH_TYPE:bddp imm[OUTPUT:CONTROLLER], cleared:true *core
28°C 多雲時晴
ENG 上午 12:56 2021/10/13

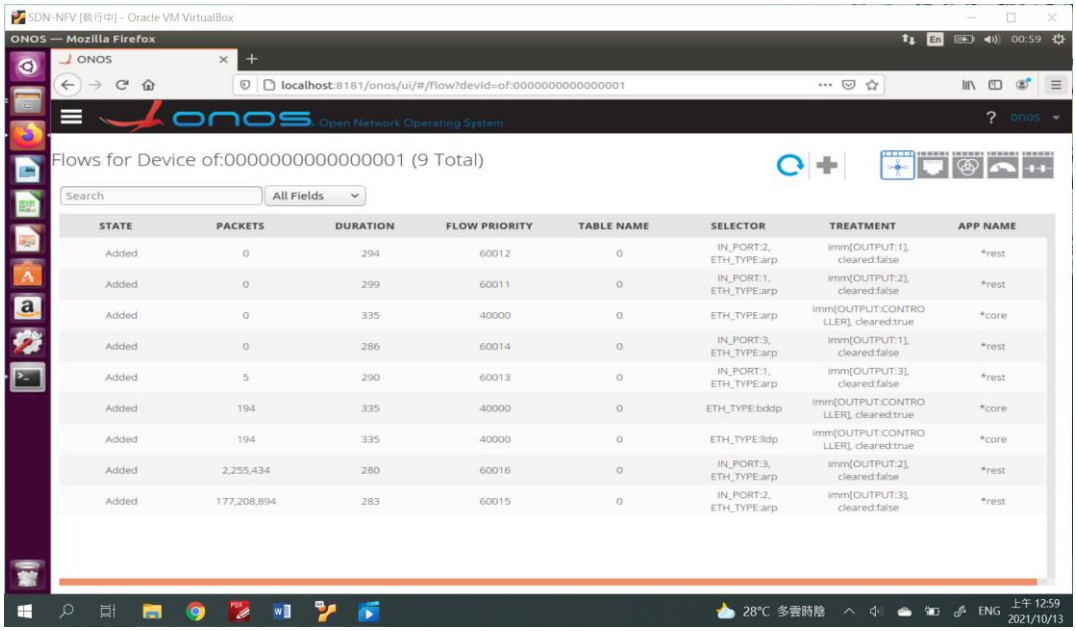
開始使用 h1 arping h2 及 h2 arping h1 後，得到以下結果(我在執行 arping 時先執行 h1 arping h2，因此下列每個 switch 的圖示中，第一張圖都只會有一個方向的 flow 有封包在傳輸，直到第二個封包執行後，相反方向的 flow 才有封包)

s1

h1 arping h2

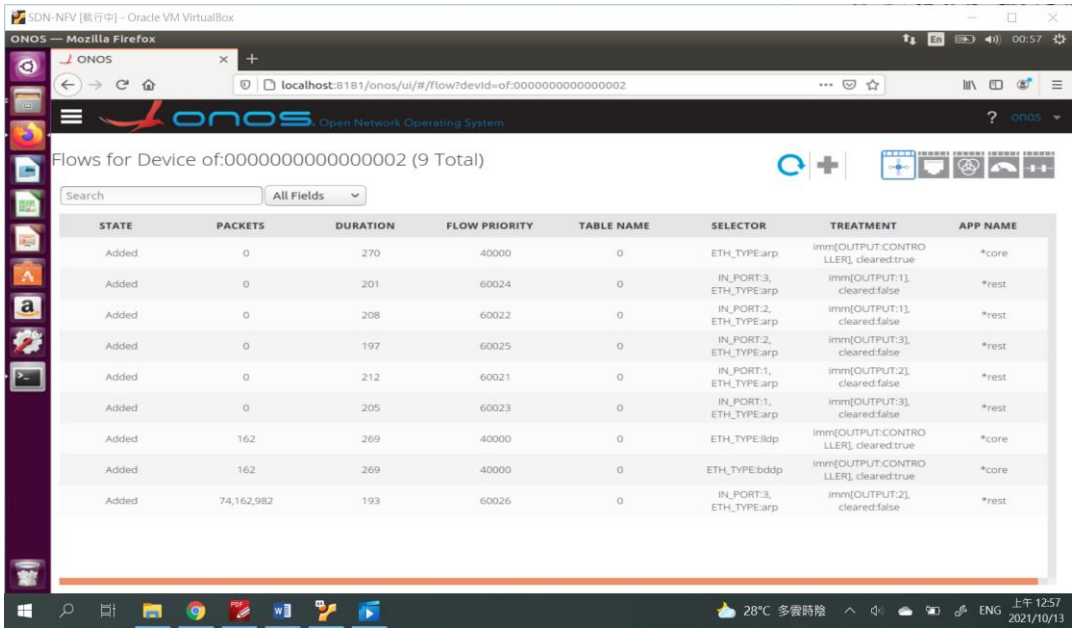


h2 arping h1

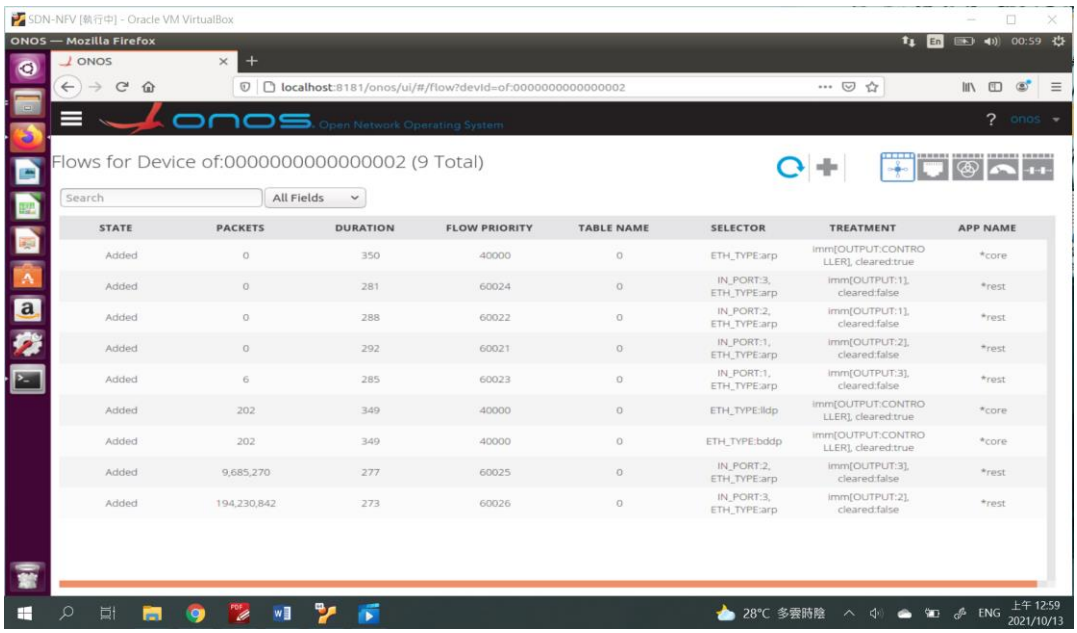


s2

h1 arping h2

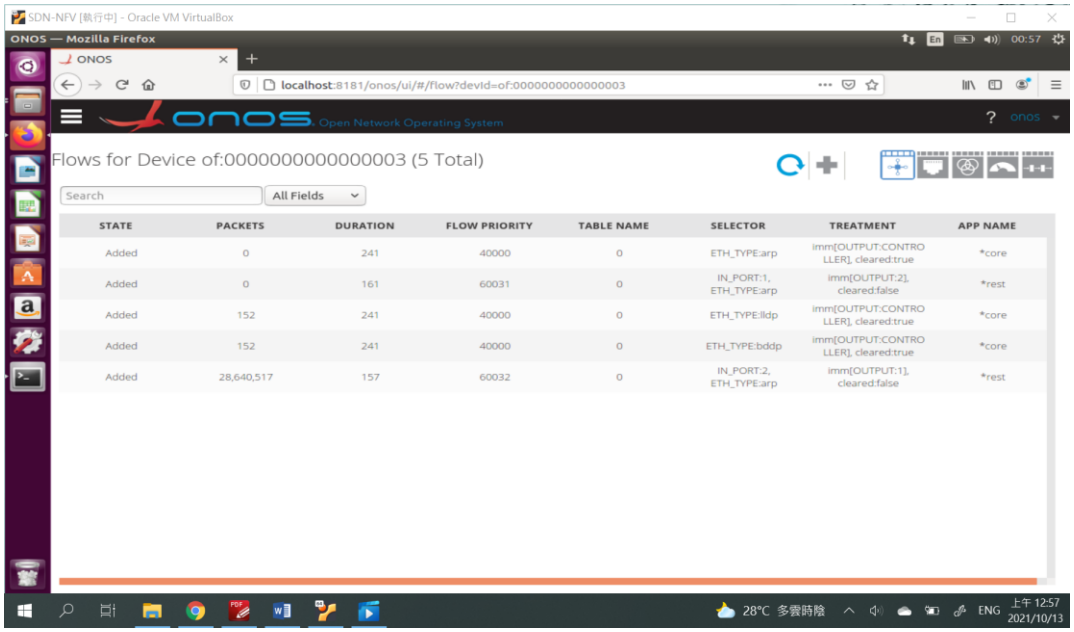


h2 arping h1

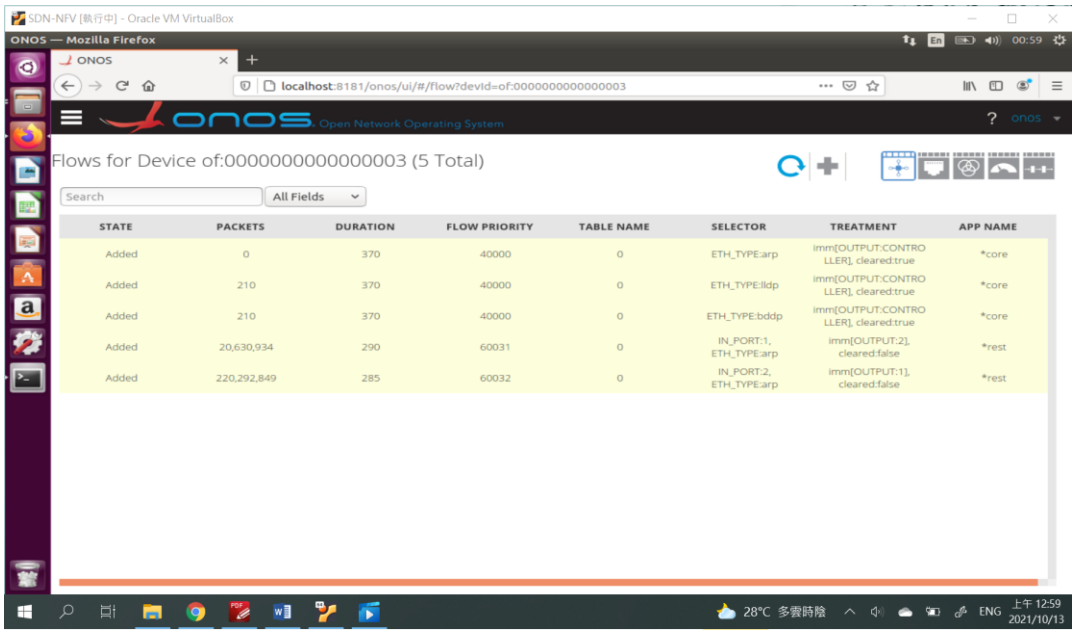


s3

h1 arping h2



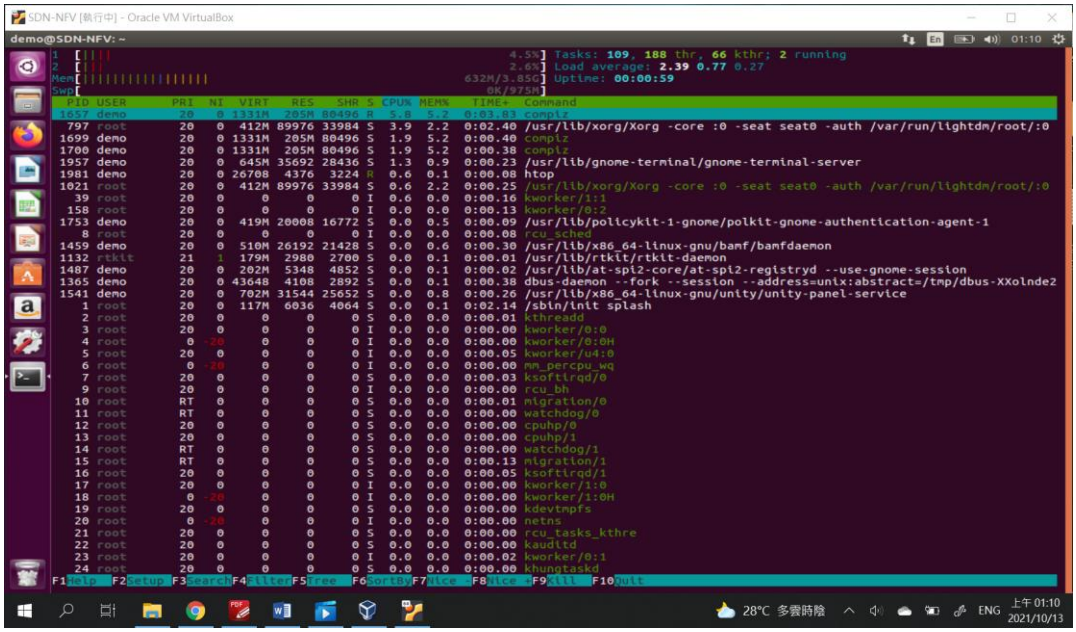
h2 arping h1



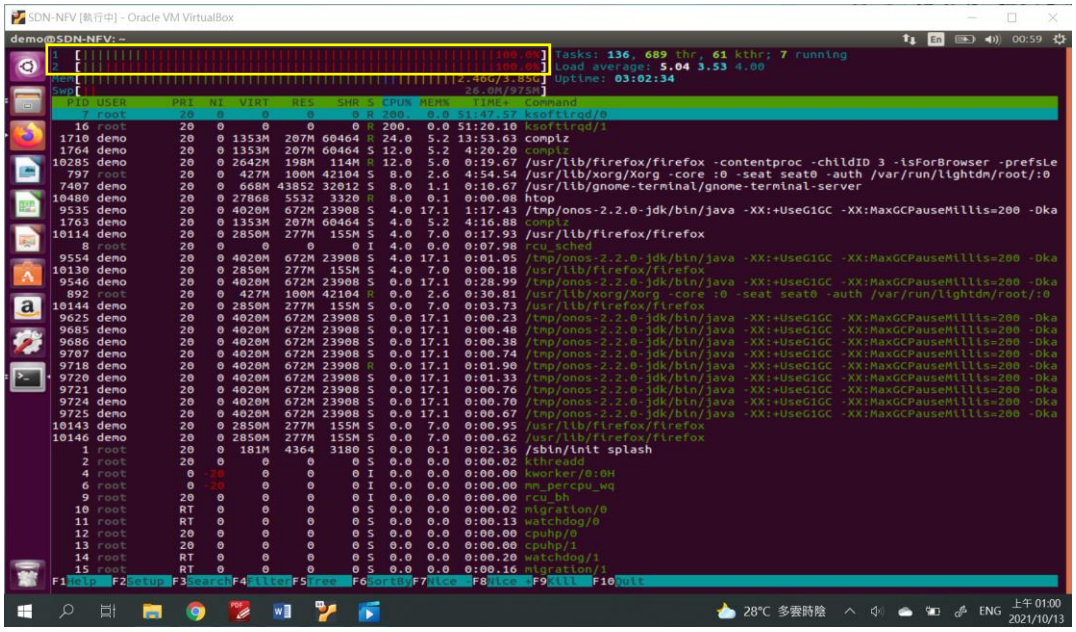
CPU 使用率

在偵測 CPU 使用率的工具方面，我選擇使用 htop，平常狀態下的 CPU 和處於 broadcast storm 下的 CPU 對比如下

一般狀態



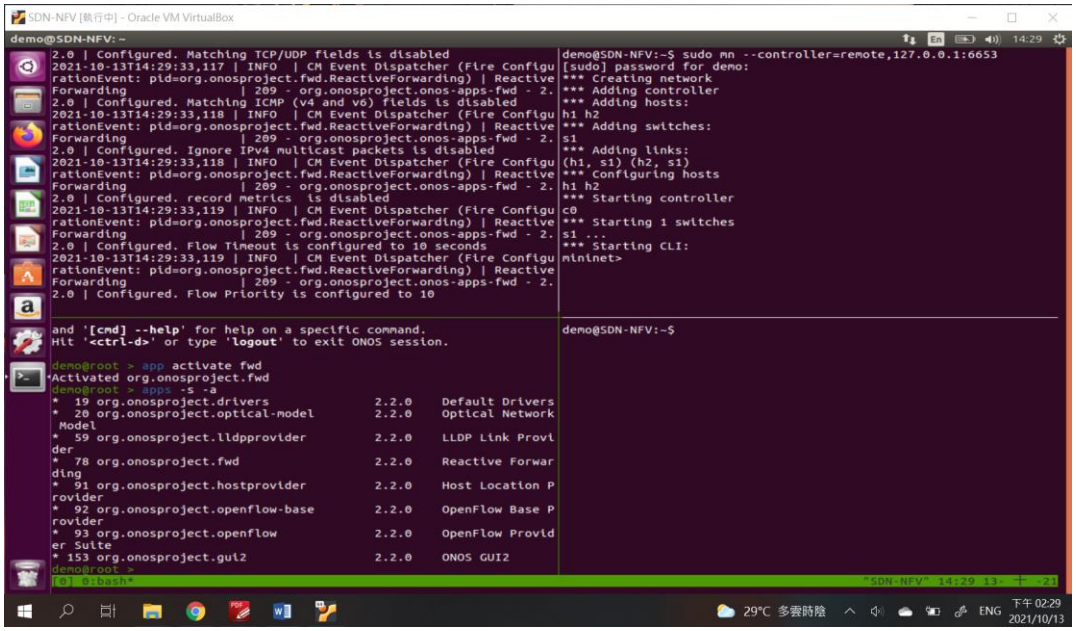
Broadcast storm



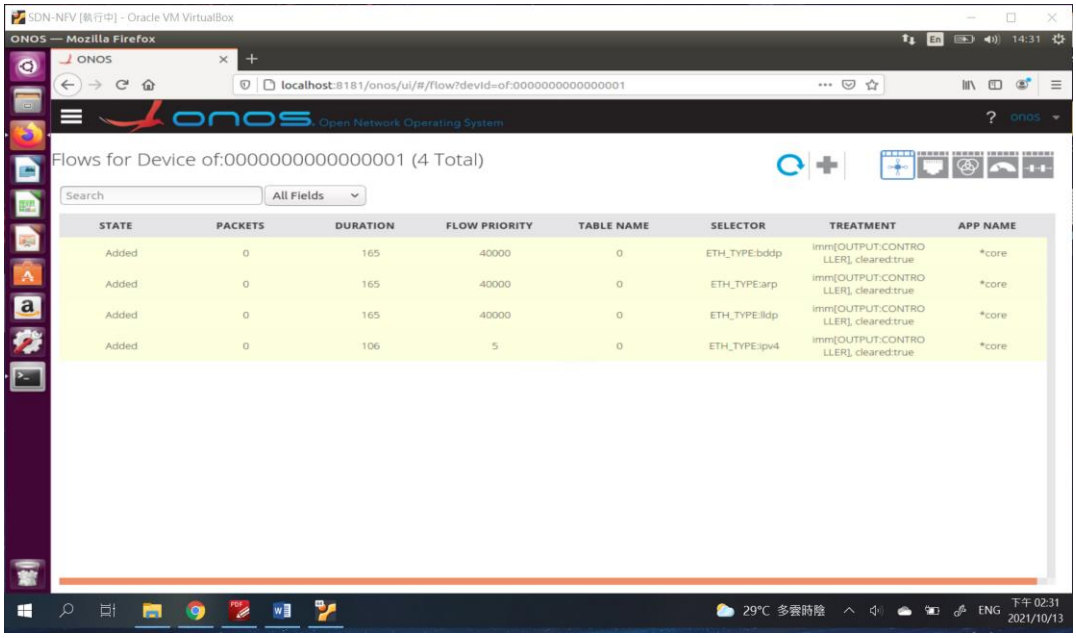
由上圖黃框處可知 2 個 CPU 都處於滿負載的情況

Part4

完成準備工作，如下圖



首先 s1 一開始不會有能讓 h1 ping h2 的 flow rule



當執行 h1 ping h2 後，封包由 h1 傳送到 s1，s1 這時候的 flow table 沒有任何的 rule 存在，因此第一個進入到 switch 的 packet 會被包裝成 openflow packet in message 並送到 ONOS，ONOS 再將訊息送到 reactive forwarding app，並使得這個 app 可以發出 flowmod 或 packetout 的訊息

Reactive forwarding app 對封包的處理如下(參考 reactive forwarding 在 github 上的 code)

Reactive forwarding 主要會執行 Reactive processor 這個 function 以完成封包的轉送和 flow rule 的設定

Reactive processor 執行步驟

1. 從 switch 抓封包上來，並將封包 parse 好

```
InboundPacket pkt = context.inPacket();
Ethernet ethPkt = pkt.parsed();
```

2. Processor 會根據自己的 function 判斷 packet 的性質(control packet、IPV6 multicast packet…)

```
MacAddress macAddress = ethPkt.getSourceMAC();
ReactiveForwardMetrics macMetrics = null;
macMetrics = createCounter(macAddress);
inPacket(macMetrics);

// Bail if this is deemed to be a control packet.
if (isControlPacket(ethPkt)) {
    droppedPacket(macMetrics);
    return;
}

// Skip IPv6 multicast packet when IPv6 forward is disabled.
if (!ipv6Forwarding && isIpv6Multicast(ethPkt)) {
    droppedPacket(macMetrics);
    return;
}

HostId id = HostId.hostId(ethPkt.getDestinationMAC());

// Do not process LLDP MAC address in any way.
if (id.mac().isLldp()) {
    droppedPacket(macMetrics);
    return;
}
```

3. 從封包中取出 source 和 destination mac address，並確認這個 switch 是否知道 packet 的 destination 在哪，如果不知道就用 flood 的方式尋找

```
HostId id = HostId.hostId(ethPkt.getDestinationMAC());


// Do we know who this is for? If not, flood and bail.
Host dst = hostService.getHost(id);
if (dst == null) {
    flood(context, macMetrics);
    return;
}
```

4. 因為我們使用的是 default topology，h2 會直接連到現在這個 switch，此時就不會使用 flood，而是經過 deviceid 和 port 的比對後，呼叫 installRule 這個 function

```
// Do we know who this is for? If not, flood and bail.
Host dst = hostService.getHost(id);
if (dst == null) {
    flood(context, macMetrics);
    return;
}

// Are we on an edge switch that our destination is on? If so,
// simply forward out to the destination and bail.
if (pkt.receivedFrom().deviceId().equals(dst.location().deviceId())) {
    if (!context.inPacket().receivedFrom().port().equals(dst.location().port())) {
        installRule(context, dst.location().port(), macMetrics);
    }
    return;
}
```

5. InstallRule 會將 packet 的 source、destination MAC address 及 port 作為篩選標準，且因為是 IPV4 message，篩選標準還會再加上 ethernet type 和 protocol

```
selectorBuilder.matchInPort(context.inPacket().receivedFrom().port())
        .matchEthSrc(inPkt.getSourceMAC())
        .matchEthDst(inPkt.getDestinationMAC());

// If configured Match Vlan ID
if (matchVlanId && inPkt.getVlanID() != Ethernet.VLAN_UNTAGGED) {
    selectorBuilder.matchVlanId(VlanId.vlanId(inPkt.getVlanID()));
}

//
// If configured and EtherType is IPv4 - Match IPv4 and
// TCP/UDP/ICMP fields
//
if (matchIpv4Address && inPkt.getEtherType() == Ethernet.TYPE_IPV4) {
    IPv4 ipv4Packet = (IPv4) inPkt.getPayload();
    byte ipv4Protocol = ipv4Packet.getProtocol();
    Ip4Prefix matchIp4SrcPrefix =
            Ip4Prefix.valueOf(ipv4Packet.getSourceAddress(),
                              Ip4Prefix.MAX_MASK_LENGTH);
    Ip4Prefix matchIp4DstPrefix =
            Ip4Prefix.valueOf(ipv4Packet.getDestinationAddress(),
                              Ip4Prefix.MAX_MASK_LENGTH);
    selectorBuilder.matchEthType(Ethernet.TYPE_IPV4)
            .matchIPSrc(matchIp4SrcPrefix)
            .matchIPDst(matchIp4DstPrefix);

    if (matchIpv4Dscp) {
        byte dscp = ipv4Packet.getDscp();
        byte ecn = ipv4Packet.getEcn();
```

```
            byte ecn = ipv4Packet.getEcn();
            selectorBuilder.matchIPDscp(dscp).matchIPEcn(ecn);
        }

        if (matchTcpUdpPorts && ipv4Protocol == IPv4.PROTOCOL_TCP) {
            TCP tcpPacket = (TCP) ipv4Packet.getPayload();
            selectorBuilder.matchIPProtocol(ipv4Protocol)
                    .matchTcpSrc(TpPort.tpPort(tcpPacket.getSourcePort()))
                    .matchTcpDst(TpPort.tpPort(tcpPacket.getDestinationPort()));
        }
        if (matchTcpUdpPorts && ipv4Protocol == IPv4.PROTOCOL_UDP) {
            UDP udpPacket = (UDP) ipv4Packet.getPayload();
            selectorBuilder.matchIPProtocol(ipv4Protocol)
                    .matchUdpSrc(TpPort.tpPort(udpPacket.getSourcePort()))
                    .matchUdpDst(TpPort.tpPort(udpPacket.getDestinationPort()));
        }
        if (matchIcmpFields && ipv4Protocol == IPv4.PROTOCOL_ICMP) {
            ICMP icmpPacket = (ICMP) ipv4Packet.getPayload();
            selectorBuilder.matchIPProtocol(ipv4Protocol)
                    .matchIcmpType(icmpPacket.getIcmpType())
                    .matchIcmpCode(icmpPacket.getIcmpCode());
        }
    }
```

6. 將 packet 送到指定的 port 並在 switch 上放置 rule

```
                        .matchIcmpv6Type(icmp6Packet.getIcmpType())
                        .matchIcmpv6Code(icmp6Packet.getIcmpCode());
                }
            }
        }
        TrafficTreatment treatment = DefaultTrafficTreatment.builder()
                .setOutput(portNumber)
                .build();

        ForwardingObjective forwardingObjective = DefaultForwardingObjective.builder()
                .withSelector(selectorBuilder.build())
                .withTreatment(treatment)
                .withPriority(flowPriority)
                .withFlag(ForwardingObjective.Flag.VERSATILE)
                .fromApp(appId)
                .makeTemporary(flowTimeout)
                .add();

        flowObjectiveService.forward(context.inPacket().receivedFrom().deviceId(),
                                     forwardingObjective);
        forwardPacket(macMetrics);
```

之後 packet 就進入 data plane 並送到 h2，然後 h2 會回傳 h1 一個 ICMP reply，因為 h2 到 h1 的 flow 並沒有在 flow table 中，上面的 control plane 流程會再走一次然後這個封包才會回傳到 h1

如此可以看到 s1 的 flow table 中有 h1 到 h2 方向及 h2 到 h1 方向的 flow rule