

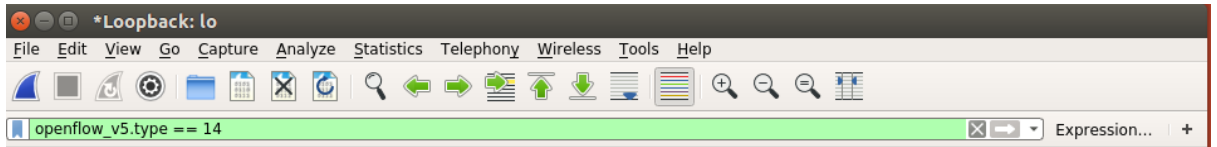
# SDN-NFV

0516045 張凱翔

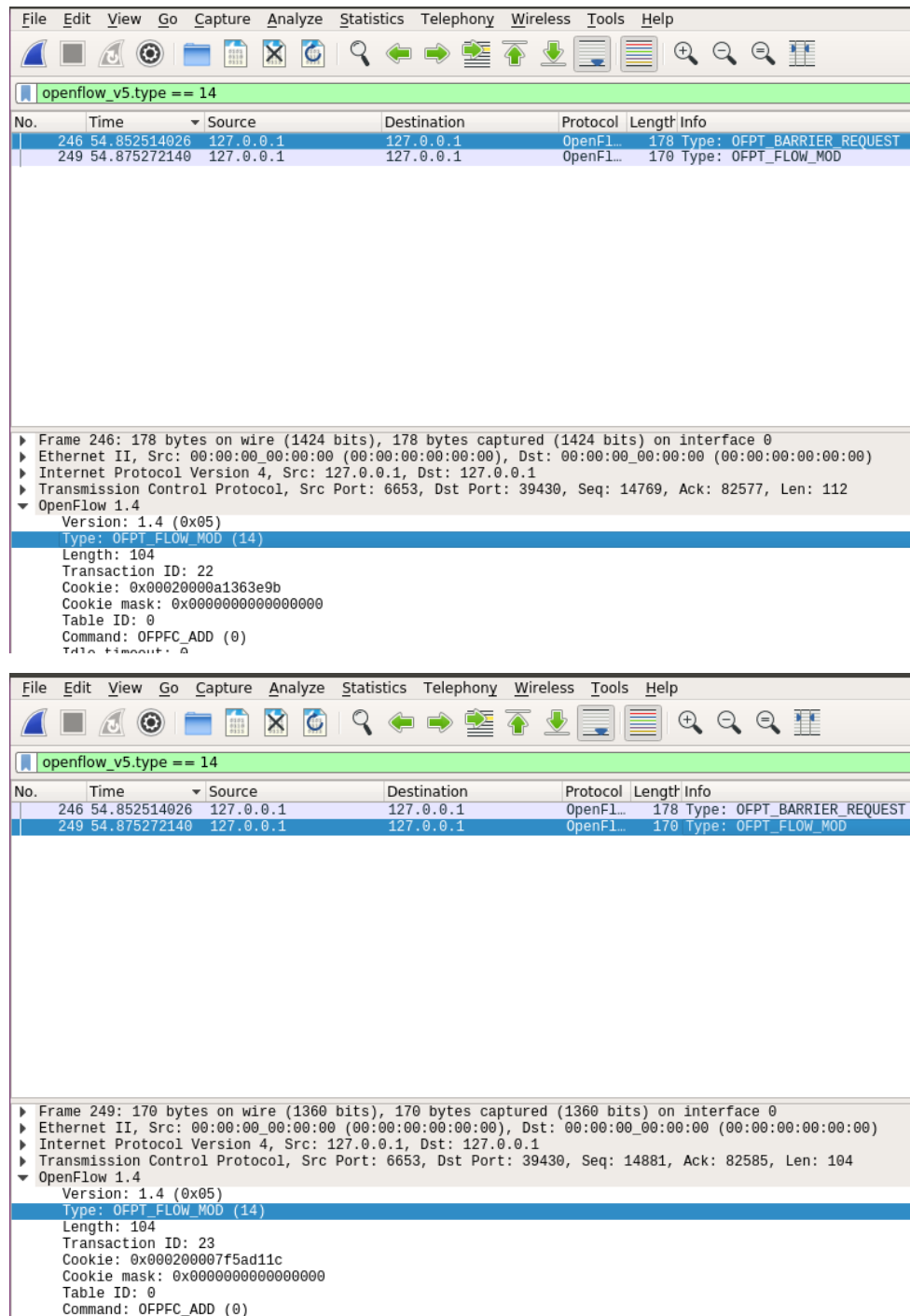
## Part1.

### 1. 2 個

將 filter 設成 `openflow_v5.type == 14`(type14 = OFPT\_FLOW\_MOD)



可以看到以下的結果



2.

a.

Wireshark capture of OpenFlow v5 traffic. The filter is `openflow_v5.type == 14`. The packet list shows two packets: packet 246 (Type: OFPT\_BARRIER\_REQUEST) and packet 249 (Type: OFPT\_FLOW\_MOD).

No.	Time	Source	Destination	Protocol	Length	Info
246	54.852514026	127.0.0.1	127.0.0.1	OpenFl...	178	Type: OFPT_BARRIER_REQUEST
249	54.875272140	127.0.0.1	127.0.0.1	OpenFl...	170	Type: OFPT_FLOW_MOD

Packet 249 details:

- Out group: OFPG\_ANY (4294967295)
- Flags: 0x0001
- Importance: 0
- Match
  - Type: OFPMT\_OXM (1)
  - Length: 32
  - OXM field
  - OXM field
  - OXM field
- Instruction
  - Type: OFPIT\_APPLY\_ACTIONS (4)
  - Length: 24
  - Pad: 00000000
  - Action
    - Type: OFPAT\_OUTPUT (0)
    - Length: 16
    - Port: 1
    - Max length: 0
    - Pad: 000000000000

Wireshark capture of OpenFlow v5 traffic. The filter is `openflow_v5.type == 14`. The packet list shows two packets: packet 246 (Type: OFPT\_BARRIER\_REQUEST) and packet 249 (Type: OFPT\_FLOW\_MOD).

No.	Time	Source	Destination	Protocol	Length	Info
246	54.852514026	127.0.0.1	127.0.0.1	OpenFl...	178	Type: OFPT_BARRIER_REQUEST
249	54.875272140	127.0.0.1	127.0.0.1	OpenFl...	170	Type: OFPT_FLOW_MOD

Packet 249 details:

- Out group: OFPG\_ANY (4294967295)
- Flags: 0x0001
- Importance: 0
- Match
  - Type: OFPMT\_OXM (1)
  - Length: 32
  - OXM field
  - OXM field
  - OXM field
- Instruction
  - Type: OFPIT\_APPLY\_ACTIONS (4)
  - Length: 24
  - Pad: 00000000
  - Action
    - Type: OFPAT\_OUTPUT (0)
    - Length: 16
    - Port: 2
    - Max length: 0
    - Pad: 000000000000

b. 10

No.	Time	Source	Destination	Protocol	Length	Info
246	54.852514026	127.0.0.1	127.0.0.1	OpenFl...	178	Type: OFPT_BARRIER_REQUEST
249	54.875272140	127.0.0.1	127.0.0.1	OpenFl...	170	Type: OFPT_FLOW_MOD

Packet 249 details:

- Type: OFPT\_FLOW\_MOD (14)
- Length: 104
- Transaction ID: 22
- Cookie: 0x00020000a1363e9b
- Cookie mask: 0x0000000000000000
- Table ID: 0
- Command: OFPFC\_ADD (0)
- Idle timeout: 0
- Hard timeout: 0
- Priority: 10

## Part2.

### 有開啟的功能

```
demo@root > apps -a -s 22:22:
* 20 org.onosproject.hostprovider 2.2.0 Host Location Provider
* 21 org.onosproject.lldpprovider 2.2.0 LLDP Link Provider
* 22 org.onosproject.optical-model 2.2.0 Optical Network Model
* 23 org.onosproject.openflow-base 2.2.0 OpenFlow Base Provider
* 24 org.onosproject.openflow 2.2.0 OpenFlow Provider Suite
* 48 org.onosproject.drivers 2.2.0 Default Drivers
* 128 org.onosproject.gui2 2.2.0 ONOS GUI2
```

### Before

Install flow rule 之前 h1 arping h2 和 h1 ping h2 都是 ping 不到的

```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
^CSent 3 probes (3 broadcast(s))
Received 0 response(s)
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms
```

### After

Install 3 個 flow rule 之後 就都可以 ping 到了

```
demo@demo-VirtualBox:~/SDN_project2/part2$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s1-1_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
demo@demo-VirtualBox:~/SDN_project2/part2$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s1-2_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
demo@demo-VirtualBox:~/SDN_project2/part2$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s1-3_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
```

```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [0E:9E:9A:43:F6:8C] 0.533ms
Unicast reply from 10.0.0.2 [0E:9E:9A:43:F6:8C] 0.607ms
Unicast reply from 10.0.0.2 [0E:9E:9A:43:F6:8C] 0.607ms
^CSent 3 probes (1 broadcast(s))
Received 3 response(s)
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.534 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.061 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.053/0.216/0.534/0.224 ms
```

## JSON 檔

### For arping

```
flows_s1-1_0516045.json x flows_s1-2_0516045.json
{
  "priority": 40001,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      },
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  }
}
```

### For IPv4

```
flows_s1-1_0516045.json x flows_s1-2_0516045.json
{
  "priority": 40002,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0800"
      },
      {
        "type": "IN_PORT",
        "port": "2"
      }
    ]
  }
}
```

```
flows_s1-1_0516045.json x flows_s1-2_0516045.json
{
  "priority": 40003,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0800"
      },
      {
        "type": "IN_PORT",
        "port": "1"
      }
    ]
  }
}
```

總共 install 三個 rule，其中比較大的差別是 ethtype，IPv4 是 0x800，ARP 是 0x806。

Criteria 為條件，instructions 為行為，只要符合條件，便會執行設定的動作，但要注意的是 priority 較高有優先執行權，所以當設定的時候要比原來預設的要高。

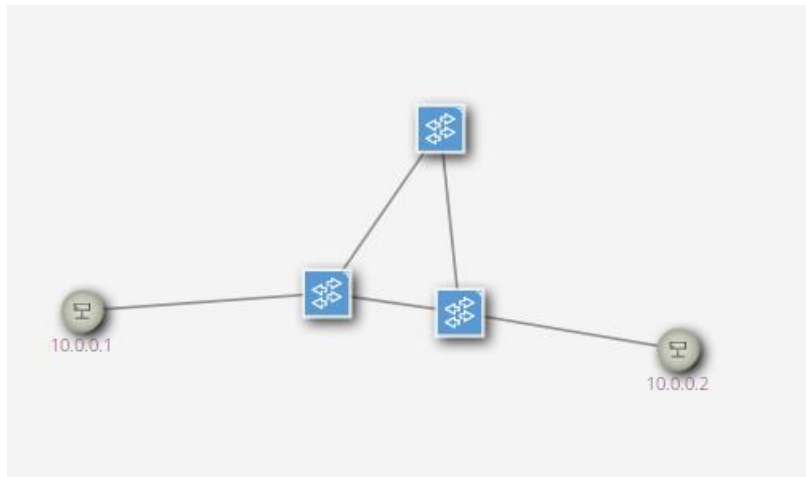
Added	0	1,334	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,334	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	5	1,000	40002	0	IN_PORT:2, ETH_TYPE:ipv4	imm[OUTPUT:1], cleared:false	*rest
Added	5	998	40003	0	IN_PORT:1, ETH_TYPE:ipv4	imm[OUTPUT:2], cleared:false	*rest
Added	6	1,334	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	16	1,002	40001	0	ETH_TYPE:arp	imm[OUTPUT:1, OUTPUT:2], cleared:false	*rest

### Part3.

#### 有開啟的功能

```
demo@root > apps -a -s 22:52
* 20 org.onosproject.hostprovider 2.2.0 Host Location Provider
* 21 org.onosproject.lldpprovider 2.2.0 LLDP Link Provider
* 22 org.onosproject.optical-model 2.2.0 Optical Network Model
* 23 org.onosproject.openflow-base 2.2.0 OpenFlow Base Provider
* 24 org.onosproject.openflow 2.2.0 OpenFlow Provider Suite
* 48 org.onosproject.drivers 2.2.0 Default Drivers
* 128 org.onosproject.gui2 2.2.0 ONOS GUI2
```

#### 建出的拓譜



#### Flow Rule

```
{
  "priority": 40001,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000002",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      },
      {
        "type": "OUTPUT",
        "port": "2"
      },
      {
        "type": "OUTPUT",
        "port": "3"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  }
}

{
  "priority": 40001,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000001",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      },
      {
        "type": "OUTPUT",
        "port": "3"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  }
}
```

```
{
  "priority": 40001,
  "timeout": 0,
  "isPermanent": true,
  "deviceId": "of:0000000000000003",
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "1"
      },
      {
        "type": "OUTPUT",
        "port": "2"
      },
      {
        "type": "OUTPUT",
        "port": "3"
      }
    ]
  },
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x0806"
      }
    ]
  }
}
```

```
demo@demo-VirtualBox:~/SDN_project2/part3$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s1-1_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
demo@demo-VirtualBox:~/SDN_project2/part3$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s2-1_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
demo@demo-VirtualBox:~/SDN_project2/part3$ curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d @flows_s3-1_0516045.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
```

## CPU Usage

可以看到當 h1 arping h2 後，CPU 的使用量直接突破天際。

Before

```
demo@demo-VirtualBox: ~/SDN_project2/part3
```

1		8.6%	Tasks: 143, 757 thr; 4 running
2		5.3%	Load average: 0.92 0.85 0.92
Mem		3.08G/3.85G	Uptime: 12:59:49
Swp		923M/975M	

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUP	MEMK	TIME+	Command
1107	root	20	0	530M	122M	38516	R	4.0	3.1	44:41.73	/usr/lib/xorg/Xor
2439	demo	20	0	603M	24448	9284	S	3.3	0.6	3:55.20	/usr/lib/gnome-te
1583	demo	20	0	1745M	209M	16388	R	2.6	5.3	2h25:57	compiz
28200	demo	20	0	4023M	642M	23104	S	2.0	16.3	2:57.94	/tmp/onos-2.2.0-j
28969	demo	20	0	34600	5160	3192	R	2.0	0.1	0:00:32	htop
1630	demo	20	0	1745M	209M	16388	S	0.7	5.3	21:53.36	compiz
1631	demo	20	0	1745M	209M	16388	S	0.7	5.3	21:49.39	compiz
1121	root	20	0	530M	122M	38516	R	0.7	3.1	1:40.72	/usr/lib/xorg/Xor
17297	root	10	-10	231M	32256	5412	S	0.7	0.8	1:41.10	ovs-vsitchd unix
7110	demo	20	0	3623M	436M	8540	S	0.7	11.1	8:36.36	bazel(onos) -XX:+
7122	demo	20	0	3623M	436M	8540	S	0.7	11.1	1:02.13	bazel(onos) -XX:+
12788	demo	20	0	3383M	367M	0	S	0.7	9.3	8:43.65	/home/demo/.cache
28206	demo	20	0	4023M	642M	23104	S	0.7	16.3	0:00.33	/tmp/onos-2.2.0-j
28394	demo	20	0	4023M	642M	23104	S	0.7	16.3	0:00.95	/tmp/onos-2.2.0-j
28395	demo	20	0	4023M	642M	23104	S	0.7	16.3	0:01.44	/tmp/onos-2.2.0-j
28423	demo	20	0	4023M	642M	23104	S	0.7	16.3	0:00.13	/tmp/onos-2.2.0-j

```
F1:1107 F2:2439 F3:1583 F4:28200 F5:28969 F6:1630 F7:1631 F8:1121 F9:17297 F10:7110
```

After

```
demo@demo-VirtualBox: ~/SDN_project2/part3
```

```
1 [|||||] 100.0% Tasks: 144, 756 thr; 5 running
2 [|||||] 100.0% Load average: 0.65 0.76 0.88
Mem [|||||] 13.09G/3.85G Uptime: 13:00:50
Swp [|||||] 923M/975M
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPUP	MEMK	TIME+	Command
1583	demo	20	0	1745M	209M	16388	R	18.0	5.3	2h26:01	compiz
28200	demo	20	0	4025M	642M	23104	S	9.0	16.3	3:01.35	/tmp/onos-2.2.0-j
1107	root	20	0	530M	122M	38516	S	7.1	3.1	44:43.27	/usr/lib/xorg/Xor
28211	demo	20	0	4025M	642M	23104	S	5.1	16.3	1:08.46	/tmp/onos-2.2.0-j
28969	demo	20	0	34600	5160	3192	R	3.9	0.1	0:02:07	htop
1630	demo	20	0	1745M	209M	16388	S	2.6	5.3	21:54.24	compiz
1631	demo	20	0	1745M	209M	16388	S	1.0	5.3	21:50.30	compiz
1172	demo	20	0	121M	0	0	S	1.3	0.0	2:08.05	/usr/bin/VBoxClie
17297	root	10	-10	231M	32256	5412	S	1.3	0.8	1:41.47	ovs-vsitchd unix
28395	demo	20	0	4025M	642M	23104	S	1.3	16.3	0:01.57	/tmp/onos-2.2.0-j
1179	demo	20	0	121M	0	0	S	0.6	0.0	2:08.04	/usr/bin/VBoxClie
28415	demo	20	0	4025M	642M	23104	S	0.6	16.3	0:00.74	/tmp/onos-2.2.0-j
2439	demo	20	0	603M	24448	9284	S	0.6	0.6	3:55.63	/usr/lib/gnome-te
1121	root	20	0	530M	122M	38516	R	0.6	3.1	1:40.80	/usr/lib/xorg/Xor
17451	demo	20	0	1541M	84804	10740	S	0.6	2.1	2:00.54	/usr/lib/firefox/
28778	demo	20	0	4025M	642M	23104	S	0.6	16.3	0:02.14	/tmp/onos-2.2.0-j
51	snmpd	51	0	100M	65K	16K	S	0.0	0.0	0:00.00	telnetd

我設計了一個有迴圈的 switch 排列，當 broadcast 時會形成一個迴圈導致 broadcast 永無止境的執行。

## **Bonus.**

這次作業花比較多時間的是 part2，花了一點時間才搞懂要怎麼把 rule 加上去，寫 rule 的時候也花了不少時間。之前就有接觸過 wireshark，但沒有深入的去研究，經過這次作業我了解了更多 wireshark 的功能，想必在這學期修的另一堂課也會有所幫助。