Part1.

1. 總共有 10 個 packets，但實際檢查 header 種類就只有 6 種。

(openflow_v5.type == 14) && (openflow_v5.flowmod.command == 0)

No.	Time	Source	Destination	Protocol	Length	Info
45	0.310184304	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
48	0.311961426	127.0.0.1	127.0.0.1	OpenFlow	274	Type: OFPT_BARRIER_REQUEST
52	0.344409262	127.0.0.1	127.0.0.1	OpenFlow	266	Type: OFPT_BARRIER_REQUEST
56	0.347955190	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
201	7.439619149	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
250	11.450056413	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
265	12.429162237	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
349	21.586672125	127.0.0.1	127.0.0.1	OpenFlow	258	Type: OFPT_FLOW_MOD
350	21.588591338	127.0.0.1	127.0.0.1	OpenFlow	258	Type: OFPT_FLOW_MOD

2.

* First header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 10
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
      .... ...0 = Has mask: False
      Length: 2
      Value: 802.1 Link Layer Discovery Protocol (LLDP) (0x88cc)
    Pad: 000000000000
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: OFPP_CONTROLLER (4294967293)
    Max length: OFPCML_NO_BUFFER (65535)
    Pad: 000000000000
```

* Second header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 10
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
      .... ...0 = Has mask: False
      Length: 2
      Value: ARP (0x0806)
    Pad: 000000000000
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: OFPP_CONTROLLER (4294967293)
    Max length: OFPCML_NO_BUFFER (65535)
    Pad: 000000000000
```

* Third header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 10
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
      .... ...0 = Has mask: False
      Length: 2
      Value: Unknown (0x8942)
    Pad: 000000000000
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: OFPP_CONTROLLER (4294967293)
    Max length: OFPCML_NO_BUFFER (65535)
    Pad: 000000000000
```

* Fourth header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 10
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 101. = Field: OFPXMT_OFB_ETH_TYPE (5)
      .... ...0 = Has mask: False
      Length: 2
      Value: IPv4 (0x0800)
    Pad: 000000000000
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: OFPP_CONTROLLER (4294967293)
    Max length: OFPCML_NO_BUFFER (65535)
    Pad: 000000000000
```

* Fifth header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 32
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 000. = Field: OFPXMT_OFB_IN_PORT (0)
      .... ...0 = Has mask: False
      Length: 4
      Value: 2
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 011. = Field: OFPXMT_OFB_ETH_DST (3)
      .... ...0 = Has mask: False
      Length: 6
      Value: 46:39:f1:28:40:c1 (46:39:f1:28:40:c1)
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
      .... ...0 = Has mask: False
      Length: 6
      Value: f2:40:d2:e0:2b:de (f2:40:d2:e0:2b:de)
```

```
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: 1
    Max length: 0
    Pad: 000000000000
```

* Sixth header

```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 32
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 000. = Field: OFPXMT_OFB_IN_PORT (0)
      .... ...0 = Has mask: False
      Length: 4
      Value: 1
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 011. = Field: OFPXMT_OFB_ETH_DST (3)
      .... ...0 = Has mask: False
      Length: 6
      Value: f2:40:d2:e0:2b:de (f2:40:d2:e0:2b:de)
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
      .... ...0 = Has mask: False
      Length: 6
      Value: 46:39:f1:28:40:c1 (46:39:f1:28:40:c1)
```

```
▼ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: 2
    Max length: 0
    Pad: 000000000000
```

3. App Name 為 “*fwd” 的 flow rule，timeout 皆為 10，App name 為 “*core” 的 flow rule，timeout 皆為 0。

0x2800000edde621

Flow ID :	0x2800000edde621
State :	Added
Bytes :	392
Packets :	4
Duration :	5
Flow Priority :	10
Table Name :	0
App Name :	*fwd
App ID :	40
Group ID :	0x0
Idle Timeout :	10
Hard Timeout :	0
Permanent :	false

0x280000f43a48ff

Flow ID :	0x280000f43a48ff
State :	Added
Bytes :	392
Packets :	4
Duration :	17
Flow Priority :	10
Table Name :	0
App Name :	*fwd
App ID :	40
Group ID :	0x0
Idle Timeout :	10
Hard Timeout :	0
Permanent :	false

0x100007a585b6f

Flow ID :	0x100007a585b6f
State :	Added
Bytes :	0
Packets :	0
Duration :	2,115
Flow Priority :	40000
Table Name :	0
App Name :	*core
App ID :	1
Group ID :	0x0
Idle Timeout :	0
Hard Timeout :	0
Permanent :	true

0x100009465555a

Flow ID :	0x100009465555a
State :	Added
Bytes :	0
Packets :	0
Duration :	2,140
Flow Priority :	40000
Table Name :	0
App Name :	*core
App ID :	1
Group ID :	0x0
Idle Timeout :	0
Hard Timeout :	0
Permanent :	true

0x10000ea6f4b8e

Flow ID :	0x10000ea6f4b8e
State :	Added
Bytes :	588
Packets :	14
Duration :	2,160
Flow Priority :	40000
Table Name :	0
App Name :	*core
App ID :	1
Group ID :	0x0
Idle Timeout :	0
Hard Timeout :	0
Permanent :	true

0x10000021b41dc

Flow ID :	0x10000021b41dc
State :	Added
Bytes :	1,372
Packets :	14
Duration :	2,170
Flow Priority :	5
Table Name :	0
App Name :	*core
App ID :	1
Group ID :	0x0
Idle Timeout :	0
Hard Timeout :	0
Permanent :	true

Part2.

```
demo@root > apps -a -s                                                   23:4
*   6 org.onosproject.optical-model        2.2.0    Optical Network Model
*   8 org.onosproject.drivers              2.2.0    Default Drivers
*  15 org.onosproject.hostprovider         2.2.0    Host Location Provider
*  17 org.onosproject.openflow-base        2.2.0    OpenFlow Base Provider
*  90 org.onosproject.lldpprovider         2.2.0    LLDP Link Provider
* 105 org.onosproject.openflow             2.2.0    OpenFlow Provider Suite
* 155 org.onosproject.gui2                 2.2.0    ONOS GUI2
```

在 deactivate “org.onosproject.fwd” 下，執行和下圖差不多的那 4 條指令，以制訂不同的 4 種 flow rule，才可以互傳 ARP 和 IPV4 Packets。

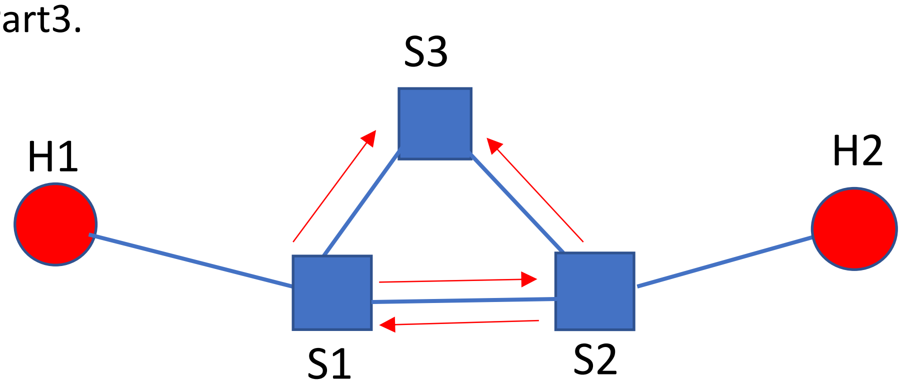
```
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Typ
e:application/json -d @flows_s1-1_310551096.json http://localh
ost:8181/onos/v1/flows/of:0000000000000001
```

Curl 完後，執行下圖兩個指令，可以成功達到 ping 的效果。

```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.627ms
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.532ms
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.529ms
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.542ms
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.530ms
Unicast reply from 10.0.0.2 [FA:AF:38:52:EF:90]  0.528ms
^CSent 6 probes (1 broadcast(s))
Received 6 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=0.167 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.027 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.030 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.031 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.025 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6145ms
rtt min/avg/max/mdev = 0.025/0.050/0.167/0.049 ms
```

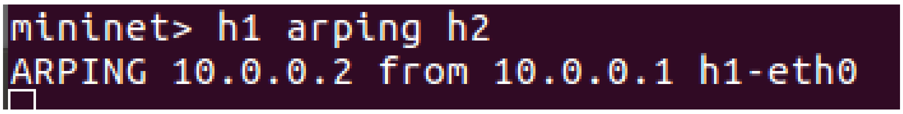
Part3.



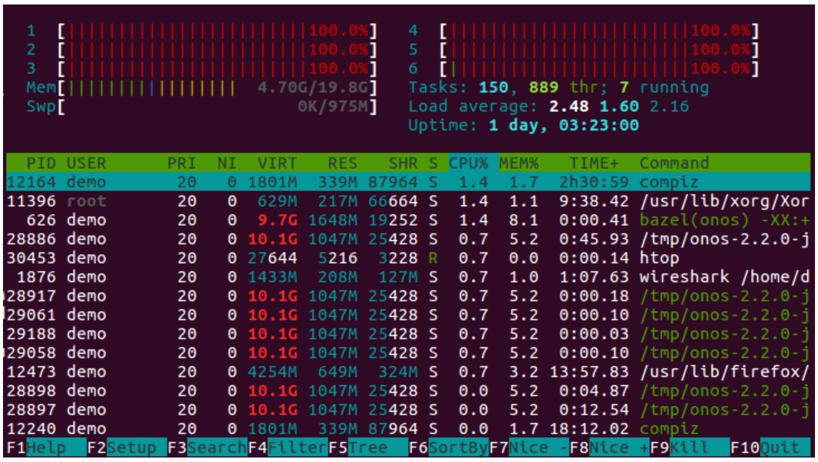
- Create a topology like the above, and create the flow rule for each 3 switches.

```
 -d @flows_s1-2_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000001
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s1-3_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000001
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s2-1_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000002
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s2-2_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000002
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s2-3_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000002
{"code":400,"message":"Invalid deviceId in flow creation request"}demo@SDN-NFV:~
 -d @flows_s2-3_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000002
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s3-1_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
000003
demo@SDN-NFV:~/Lab2$ curl -u onos:rocks -X POST -H Content-Type:application/json
 -d @flows_s3-2_310551096.json http://localhost:8181/onos/v1/flows/of:0000000000
```

- Send ARP packets from h1 to h2. Since my flow rule from S2 would not output the packet to h2(port1), so there won't have any result after I type this command.



- However, the packets are actually looping between the 3 switches. Therefore, the CPU utilization will be all 100%. In that way, there would have so called "broadcast storm".



Part4.

- Host 1 sends an ARP request to the switch first **(in data plane)**
- The switch receives the ARP request, and forwards it to the controller **(between control and data plane)**
- The controller receives the ARP request, and check its ARP table to find host 2's MAC address **(in control plane)**
- After finding the MAC address of host 2, the controller sends the ARP reply back to the switch **(between control plane and data plane)**
- The switch receives the ARP reply, and forward it back to the host 1 **(in data plane)**
- After receiving the ARP reply, host 1 would know the MAC address of host 2. Then, host1 sends an ICMP request to the switch **(in data plane)**
- The switch receives the ICMP request, and checks its MAC address with the forwarding table **(in data plane)**
- Finally, the switch sends the ICMP request to the destination, according to the MAC address => host 2 **(in data plane)**

What I've learned or solved

- The use of wireshark
- How to distinguish the different openflow header
- The difference between "arping" and "ping".
- Use the json to construct the flow rule.
- Basic operation of "curl" command.
- The definition of "Broadcast Strom"