LAB02 - OpenFlow Protocol Observation and Flow Rule Installation

310581040 智能所 簡郁宸

Part 1: Answer Questions

1. **How many OpenFlow headers with type “OFPT_FLOW_MOD” and command “OFPFC_ADD” are there among all the packets?**

   A: 共有六種 distinct type 為`OFPT_FLOW_MOD` 且 command 為 `OFPFC_ADD` 的 OpenFlow headers

   1.

   openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
195	30.484739439	127.0.0.1	127.0.0.1	OpenFlow	274	Type: OFPT_BARRIER_REQUEST
200	30.486405625	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
350	38.404457706	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
576	52.560485521	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
586	53.448261699	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

   ```
   Transmission Control Protocol, Src Port: 6653, Dst Port: 54242, Seq: 1243, Ack: 1917, Len: 208
   OpenFlow 1.4
       Version: 1.4 (0x05)
       Type: OFPT_FLOW_MOD (14)
       Length: 96
       Transaction ID: 1
       Cookie: 0x00010000ea6f4b8e
       Cookie mask: 0x0000000000000000
       Table ID: 0
       Command: OFPFC_ADD (0)
       Idle timeout: 0
       Hard timeout: 0
       Priority: 40000
       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: 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
   ```

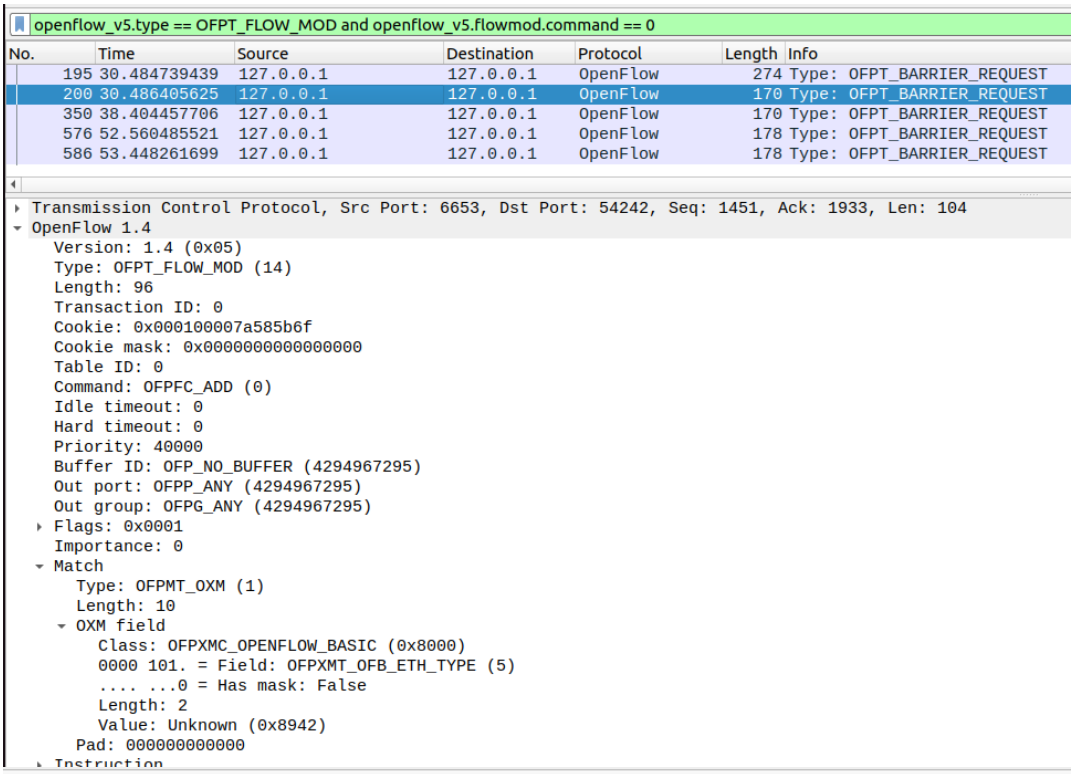
   2.

   openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
195	30.484739439	127.0.0.1	127.0.0.1	OpenFlow	274	Type: OFPT_BARRIER_REQUEST
200	30.486405625	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
350	38.404457706	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
576	52.560485521	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
586	53.448261699	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

   ```
   OpenFlow 1.4
       Version: 1.4 (0x05)
       Type: OFPT_FLOW_MOD (14)
       Length: 96
       Transaction ID: 2
       Cookie: 0x000100009465555a
       Cookie mask: 0x0000000000000000
       Table ID: 0
       Command: OFPFC_ADD (0)
       Idle timeout: 0
       Hard timeout: 0
       Priority: 40000
       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: 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
   ```

3.

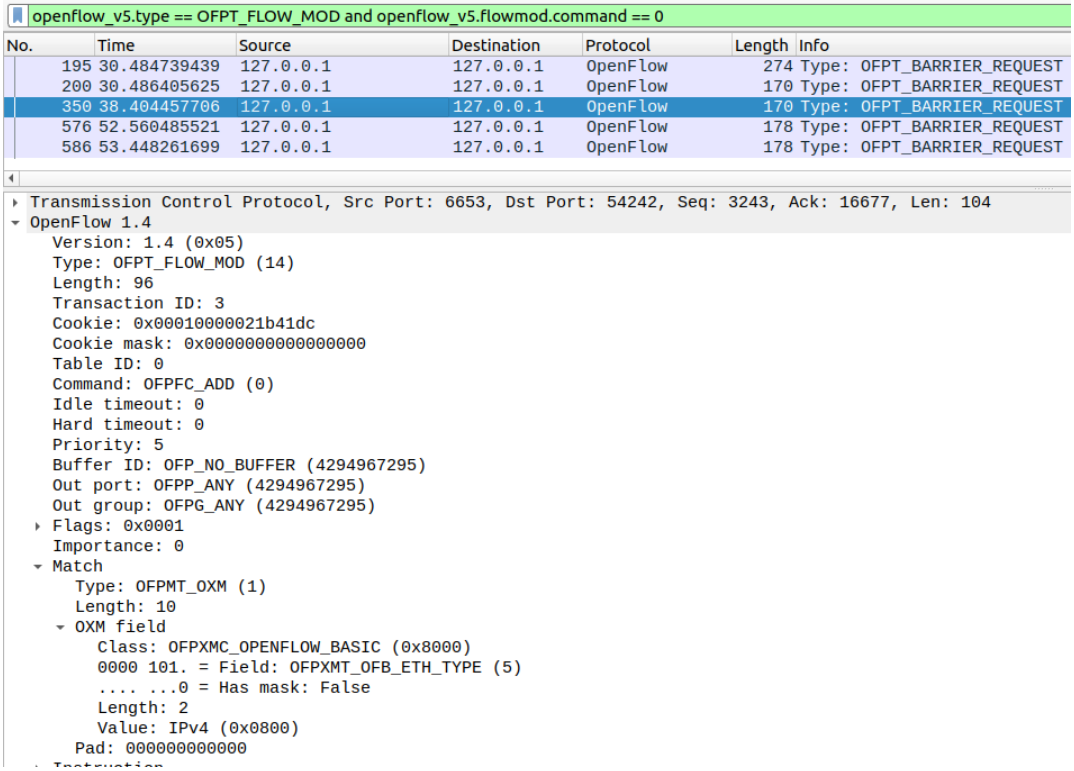


openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
195	30.484739439	127.0.0.1	127.0.0.1	OpenFlow	274	Type: OFPT_BARRIER_REQUEST
200	30.486405625	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
350	38.404457706	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
576	52.560485521	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
586	53.448261699	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

```
Transmission Control Protocol, Src Port: 6653, Dst Port: 54242, Seq: 1451, Ack: 1933, Len: 104
OpenFlow 1.4
    Version: 1.4 (0x05)
    Type: OFPT_FLOW_MOD (14)
    Length: 96
    Transaction ID: 0
    Cookie: 0x000100007a585b6f
    Cookie mask: 0x0000000000000000
    Table ID: 0
    Command: OFPFC_ADD (0)
    Idle timeout: 0
    Hard timeout: 0
    Priority: 40000
    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: 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
  ▸ Instruction
```

4.



openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
195	30.484739439	127.0.0.1	127.0.0.1	OpenFlow	274	Type: OFPT_BARRIER_REQUEST
200	30.486405625	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
350	38.404457706	127.0.0.1	127.0.0.1	OpenFlow	170	Type: OFPT_BARRIER_REQUEST
576	52.560485521	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST
586	53.448261699	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

```
Transmission Control Protocol, Src Port: 6653, Dst Port: 54242, Seq: 3243, Ack: 16677, Len: 104
OpenFlow 1.4
    Version: 1.4 (0x05)
    Type: OFPT_FLOW_MOD (14)
    Length: 96
    Transaction ID: 3
    Cookie: 0x00010000021b41dc
    Cookie mask: 0x0000000000000000
    Table ID: 0
    Command: OFPFC_ADD (0)
    Idle timeout: 0
    Hard timeout: 0
    Priority: 5
    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: 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
  ▸ Instruction
```

5. openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
576	52.560485521	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

```
OpenFlow 1.4
    Version: 1.4 (0x05)
    Type: OFPT_FLOW_MOD (14)
    Length: 104
    Transaction ID: 6
    Cookie: 0x008d00000b66b43e
    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)
    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_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: 5a:64:4c:3d:47:52 (5a:64:4c:3d:47:52)
    ▾ OXM field
        Class: OFPXMC_OPENFLOW_BASIC (0x8000)
        0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
        .... ...0 = Has mask: False
        Length: 6
        Value: ce:f3:b6:14:9b:5c (ce:f3:b6:14:9b:5c)
```

6. openflow_v5.type == OFPT_FLOW_MOD and openflow_v5.flowmod.command == 0

No.	Time	Source	Destination	Protocol	Length	Info
586	53.448261699	127.0.0.1	127.0.0.1	OpenFlow	178	Type: OFPT_BARRIER_REQUEST

```
OpenFlow 1.4
    Version: 1.4 (0x05)
    Type: OFPT_FLOW_MOD (14)
    Length: 104
    Transaction ID: 7
    Cookie: 0x008d0000f9cadeb4
    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)
    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_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: ce:f3:b6:14:9b:5c (ce:f3:b6:14:9b:5c)
    ▾ OXM field
        Class: OFPXMC_OPENFLOW_BASIC (0x8000)
        0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
        .... ...0 = Has mask: False
        Length: 6
        Value: 5a:64:4c:3d:47:52 (5a:64:4c:3d:47:52)
```

2. **What are the match fields and the corresponding actions in each “OFPT_FLOW_MOD” message?**
3. **What are the Idle Timeout values for all flow rules on s1 in GUI?**

Match Fields	Actions	Timeout Values
ETH_TYPE = ARP	OUTPUT = CONTROLLER	0
ETH_TYPE = 802.1 Link Layer Discovery Protocol (LLDP)	OUTPUT = CONTROLLER	0
ETH_TYPE = Unknown (0x8942) **(bddp)**	OUTPUT = CONTROLLER	0
ETH_TYPE = IPv4	OUTPUT = CONTROLLER	0
IN_PORT=2, ETH_DST=5a:64:4c:3d:47:52, ETH_SRC=ce:f3:b6:14:9b:5c	OUTPUT = 1	10
IN_PORT=1, ETH_DST=ce:f3:b6:14:9b:5c, ETH_SRC=5a:64:4c:3d:47:52	OUTPUT = 2	10

Part 2: Install Flow Rules

> IN_PORT -> Ingress port Numerical representation of incoming port, starting at 1. This may be a physical or switch-defined logical port.
>
> OUTPUT -> The Output action forwards a packet to a specified OpenFlow port (see 4.1). OpenFlow switches must support forwarding to physical ports, switch-defined logical ports and the required reserved ports (see 4.5).

Topology

```
sudo mn --controller=remote,127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
```

Install one flow rule to forward ARP packets

- Match Fields
  - Ethernet type (ARP)
- Actions
  - Forwarding ARP packets to all port in one instruction

```
{
  "priority": 50001,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x806"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "ALL"
      }
    ]
  }
}
```

- Take **screenshot** to verify the flow rules you installed

```
mininet> h1 arping h2
```

```
mininet> h1 arping h2
ARPING 10.0.0.2
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=0 time=162.338 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=1 time=6.998 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=2 time=5.823 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=3 time=5.548 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=4 time=5.616 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=5 time=6.045 usec
42 bytes from 4a:0b:d3:11:08:2b (10.0.0.2): index=6 time=3.587 usec
```

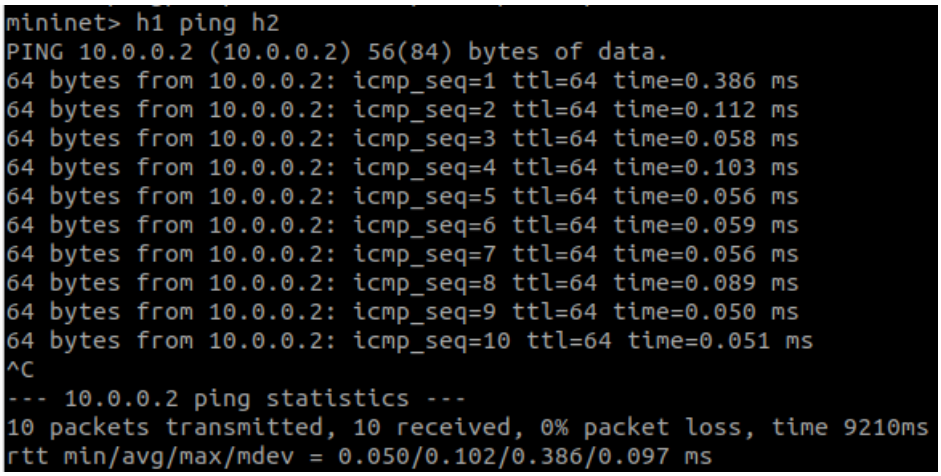
Install two flow rules to forward IPv4 packets

- Match Fields
  - IPv4 destination address and other required dependencies
- Actions
  - Forwarding IPv4 packets to the right host

```
{
  "priority": 50002,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x800"
      },
      {
        "type": "IPV4_SRC",
        "ip": "10.0.0.0/8"
      },
      {
        "type": "IPV4_DST",
        "ip": "10.0.0.0/8"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "ALL"
      }
    ]
  }
}
```

- Take **screenshot** to verify the flow rules you installed

```
mininet> h1 ping h2
```



Part 3: Create Topology with Broadcast Storm

- Steps:
  - Create a topology that may cause a “Broadcast Storm”.

    ```
    sudo mn --custom=topo_310581040.py --topo=topo_310581040 --
    controller=remote,127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
    ```

  - Install flow rules on switches.

```
curl -v -u onos:rocks  -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d @flows_s1-1_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000001'
curl -v -u onos:rocks  -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d @flows_s1-2_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000001'

curl -v -u onos:rocks -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'  -d @flows_s2-1_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000002'
curl -v -u onos:rocks -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'  -d @flows_s2-2_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000002'
curl -v -u onos:rocks -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'  -d @flows_s2-3_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000002'

curl -v -u onos:rocks -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'  -d @flows_s3-1_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000003'
curl -v -u onos:rocks -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'  -d @flows_s3-2_310581040.json
'http://localhost:8181/onos/v1/flows/of:0000000000000003'
```

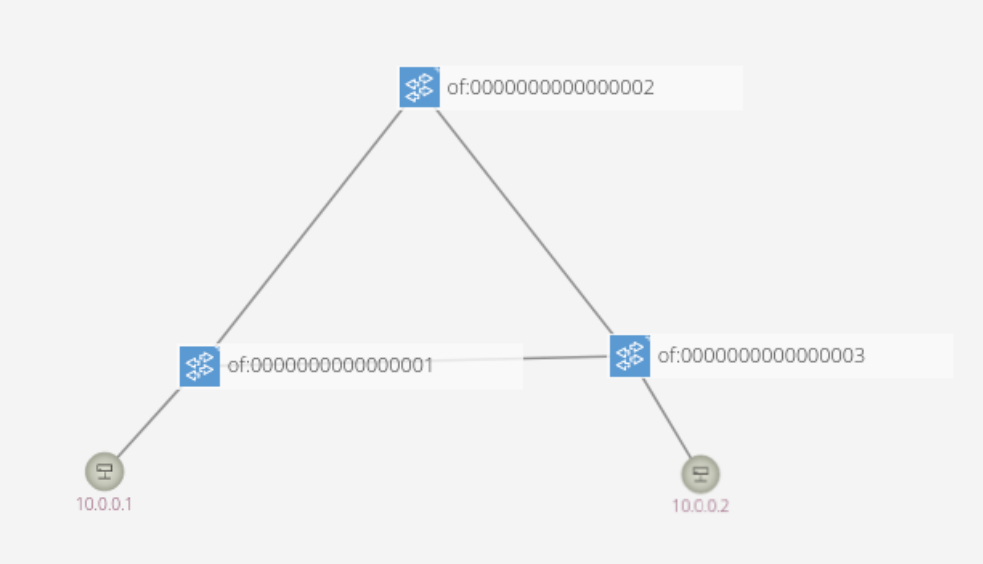
- Send packets from one host to another host.

```
h1 ping h2
```

- Observe link status of the network and the CPUs utilization of VM

- Describe what you have observed and explain why the broadcast storm occurred
  - broadcast storm 設計方式
    - Topology

      設計三個 switch 並且有 其中 s1 和 s3 連接 h1 和 h2

    - Flow Rule

      沿用 part 2 的實作

- 因為要確保每個 host 知道彼此的 IP 位置所對應的MAC, 所以需要先建立 ARP 傳輸，OUTPUT 設為 ALL 主要是希望可以在任何的 port 都可以接收到怎麼到另一個 host 的位置

```
{
  "priority": 50001,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x806"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "ALL"
      }
    ]
  }
}
```

- s1 和 s3 分別建立，當IPv4_SRC 或 IPv4_DIST 為 10.0.0.0/8 則會 OUTPUT 非 ingress 的 port (也就是會有多個 packet 輸出)

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x800"
      },
      {
        "type": "IPV4_SRC",
        "ip": "10.0.0.0/8"
      },
      {
        "type": "IPV4_DST",
        "ip": "10.0.0.0/8"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "ALL"
```

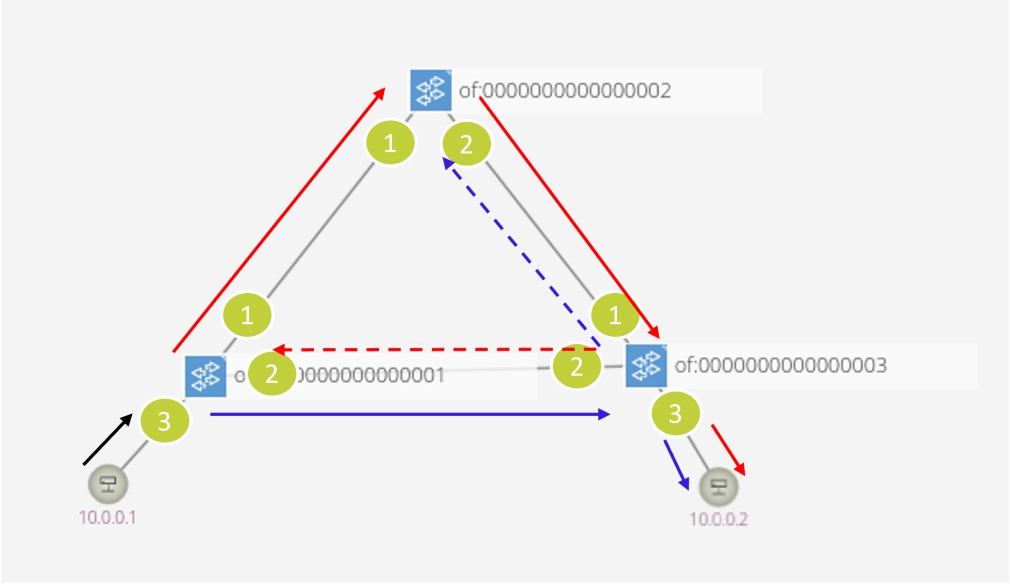
```
      }
    ]
  }
}
```

- s2 分別定義 IN_PORT 怎麼轉發

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "IN_PORT",
        "port": "1"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "2"
      }
    ]
  }
}
```

- 產生 broadcast storm 原因

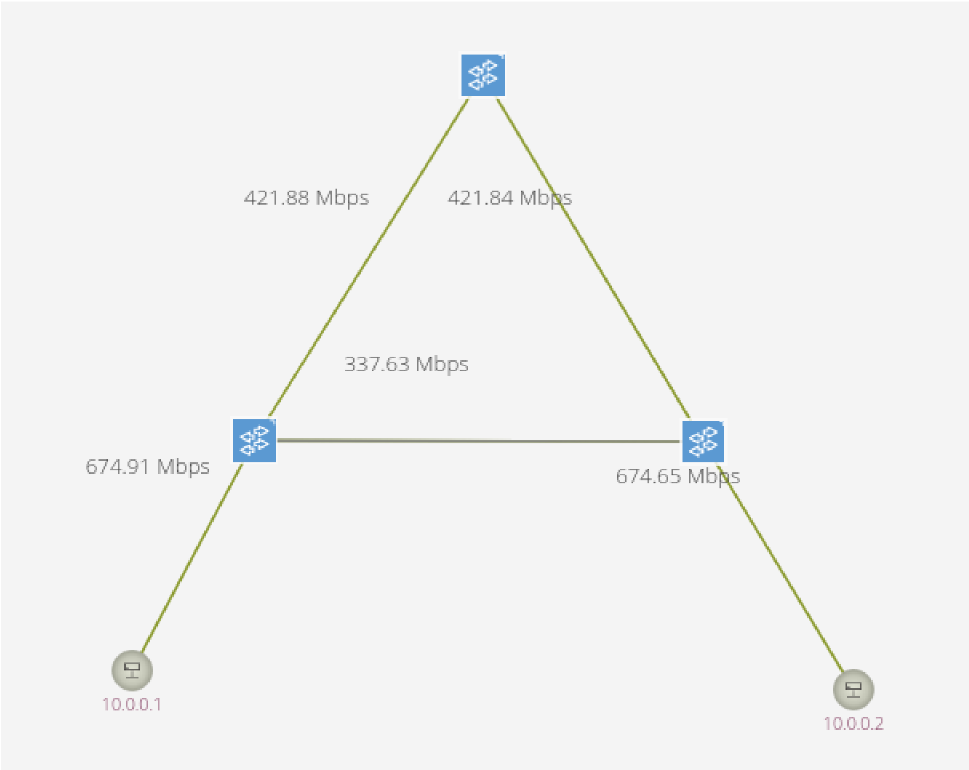
  當 h1 傳送封包時，因為 OUTPUT 為 ALL 所以會傳輸兩條路徑(分別為紅色和藍色線)，接著當傳輸至 s3 時，因為 s3 的flow rule 也是 OUTPUT ALL 所以當接收到 packet 時會再回頭傳送導致 s1 會再收到相同的 packet 而產生 loop (虛線)。



- 當遇到 broadcast storm 的情況時觀察的現象
  - link status

    會有多個重複的封包不斷在 topology 中打轉產生 loop，且傳送速率以百萬為每秒計算。

    當 `h1 ping h2` 時，會出現 (DUP!) 的字樣，表示產生速度過快導致



Flows for Device of:0000000000000001 (5 Total)

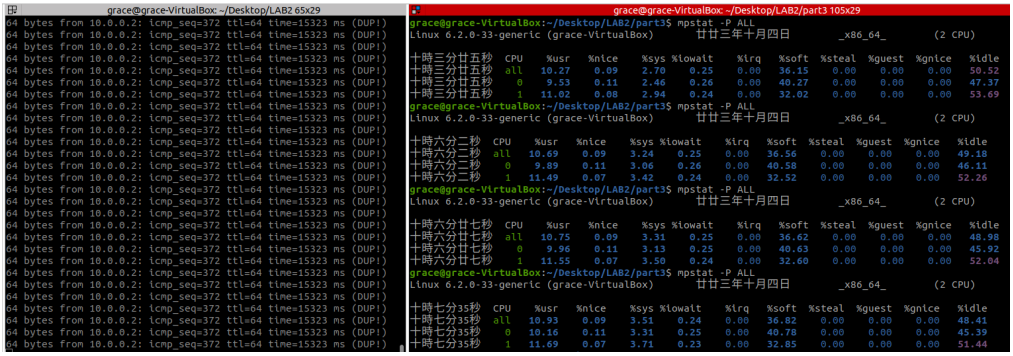
STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	4	270	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	221	270	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	221	270	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,671,943	152	50001	0	ETH_TYPE:arp	imm[OUTPUT:ALL], cleared:false	*rest
Added	91,113,685	152	50000	0	ETH_TYPE:ipv4, IPV4_SRC:10.0.0.0/8, IPV4_DST:10.0.0.0/8	imm[OUTPUT:ALL], cleared:false	*rest

```
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1726 ms (DUP!)

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, +75893 duplicates, 0% packet loss, time 999m
s
rtt min/avg/max/mdev = 1.910/819.007/1732.023/509.522 ms, pipe 2
```

  - CPUs utilization of VM

    隨著 host 之間傳送 packet 的時間越長， CPU idle state 的比例不斷下降(右圖)

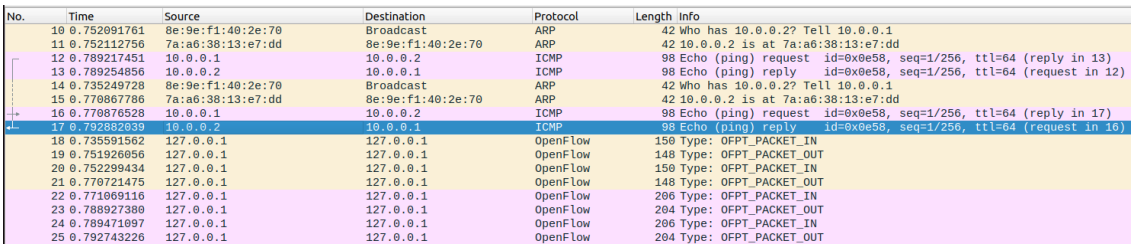


Part 4: Trace Reactive Forwarding

- Activate only “org.onosproject.fwd” and other initially activated APPs.
- Use Mininet default topology and let h1 ping h2.

```
sudo mn --controller=remote,127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
```

- Observe what happens in control and data planes
  - From the time when h1 pings h2 until h2 receives the first ICMP request
  - Write down each operation made by control and data planes
  - Please refer to the ONOS Reactive Forwarding application -- Source Code
- Describe what you observed step by step in report



No.	Time	Source	Destination	Protocol	Length	Info
10	0.752091761	8e:9e:f1:40:2e:70	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
11	0.752112756	7a:a6:38:13:e7:dd	8e:9e:f1:40:2e:70	ARP	42	10.0.0.2 is at 7a:a6:38:13:e7:dd
12	0.789217451	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x0e58, seq=1/256, ttl=64 (reply in 13)
13	0.789254856	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x0e58, seq=1/256, ttl=64 (request in 12)
14	0.735249728	8e:9e:f1:40:2e:70	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1
15	0.770867786	7a:a6:38:13:e7:dd	8e:9e:f1:40:2e:70	ARP	42	10.0.0.2 is at 7a:a6:38:13:e7:dd
16	0.770876528	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x0e58, seq=1/256, ttl=64 (reply in 17)
17	0.792882039	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x0e58, seq=1/256, ttl=64 (request in 16)
18	0.735591562	127.0.0.1	127.0.0.1	OpenFlow	150	Type: OFPT_PACKET_IN
19	0.751926056	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_PACKET_OUT
20	0.752299434	127.0.0.1	127.0.0.1	OpenFlow	150	Type: OFPT_PACKET_IN
21	0.770721475	127.0.0.1	127.0.0.1	OpenFlow	148	Type: OFPT_PACKET_OUT
22	0.771069116	127.0.0.1	127.0.0.1	OpenFlow	206	Type: OFPT_PACKET_IN
23	0.788927380	127.0.0.1	127.0.0.1	OpenFlow	204	Type: OFPT_PACKET_OUT
24	0.789471097	127.0.0.1	127.0.0.1	OpenFlow	206	Type: OFPT_PACKET_IN
25	0.792743226	127.0.0.1	127.0.0.1	OpenFlow	204	Type: OFPT_PACKET_OUT

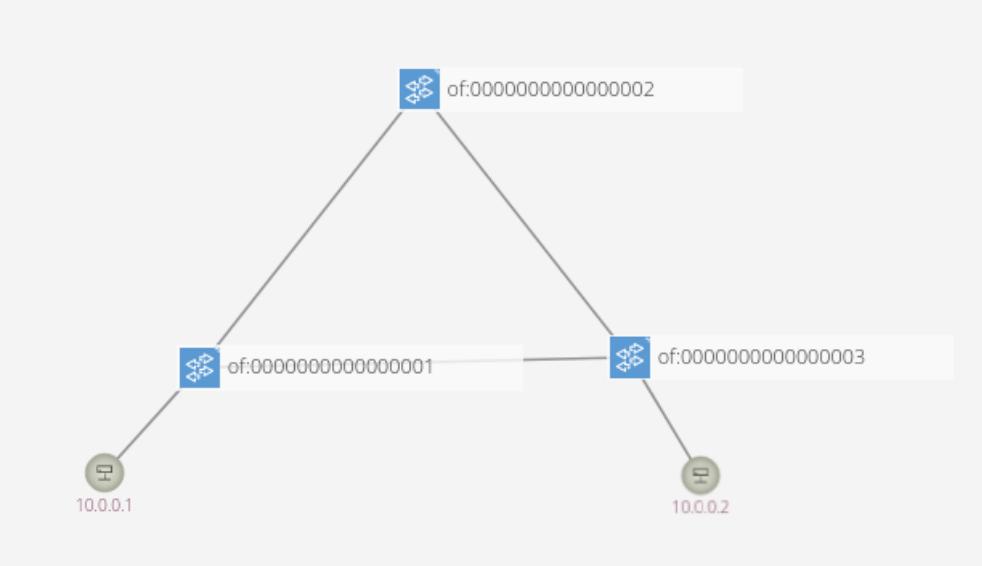
1. h1丟出一個 broadcast ARP Request (Who has 10.0.0.2? Tell 10.0.0.1)
2. 同時 s1 會傳送一個 Open Flow (data 為 ARP request) 去問 c1 (OFPT_PACKET_IN: IN_PORT:1)
3. c1 回傳 s1 一個 Open Flow (OFPT_PACKET_OUT: FLOOD)
4. h2 收到 h1 的 ARP Request 後會回傳 ARP reply (10.0.0.2 is at 7a:a6:38:13:e7:dd)
5. 同時 s1 也會傳送一個 Open Flow (data 為 ARP reply) 去問 c1 (OFPT_PACKET_IN: IN_PORT:2)
6. c1 回傳 s1 一個 Open Flow (OFPT_PACKET_OUT: OUTPUT:FLOOD)
7. h1 收到 h2 傳送的 ARP reply，得知 10.0.0.2 在哪個 mac，並傳送 ICMP (Echo ping request) 至 h2
8. 同時 s1 會傳送一個 Open Flow (data 為 Echo ping request) 去問 c1 (OFPT_PACKET_IN: IN_PORT:1)
9. c1 回傳 s1 (OFPT_PACKET_OUT: FLOOD)
10. h2 收到 **first ICMP request** 後會回傳 ICMP reply
11. 同時 s1 會傳送一個 Open Flow (data 為 Echo ping reply) 去問 c1 (OFPT_PACKET_IN: IN_PORT:2)
12. c1 回傳 s1 (OFPT_PACKET_OUT: FLOOD)

What you’ve learned or solved

1. app activate Activated org.onosproject.fwd 可以簡寫成 app activate fwd
2. 了解 Openflow 有 Ingress port 和 output port，分別為接收 packet 的 port 和傳送 packet 出去的 port
3. Part 2 其中一項要求是只能有一個 instruction 去傳輸 ARP，不能單純只是使用 IN_PORT 和 OUTPUT 一個特別的 port ，特別去查 SPEC ，找出可使用 `ALL` 來做 broadcast 的操作，有利 Part 3 的呈現
4. Part 2 中的 ipv4_src 和 ipv4_dst 輸入的是一個網域的 ip，非特定某個 host 的 ip，所以在 OUTPUT 時 直接使用 ALL 而非指定 port 可以直接讓兩組 host 可以 ping 的到
5. `curl` RESTFUL GET 時呈現的 json 很醜，所以特別查了一下怎麼顯現 pretty json (json_pp)
6. 第三部分中思考了蠻久的，首先專有名詞 `Broadcast Storm` 沒有很清楚是甚麼，所以上網去了解情境，再到後面的設計沿用了 part 2 的 code 使用 ipv4 來傳輸，因為在設計 json 的時候卡關蠻久的，其中一個發現是若沒有先建立 ARP 傳輸時，IPv4 會無法傳輸，應該是要先知道其他人的 ip 連接在哪裡。
7. 另外，也有觀察到當 `fwd` app activate 且 h1 ping h2 時產生的 flow role 是依據 MAC 的位址去做傳輸的 。

Observation and Question

在下面中的情境



在未設計 Rule 時 default link 如下

Links (3 total)

	PORT 1	PORT 2	TYPE	DIRECTION
✔	of:0000000000000002/2	of:0000000000000003/1	Direct	A ↔ B
✔	of:0000000000000002/1	of:0000000000000001/1	Direct	A ↔ B
✔	of:0000000000000001/2	of:0000000000000003/2	Direct	A ↔ B

若設計的 s2 flow rule 只有單一方向 IN_PORT = 1, OUTPUT = 2 (沒有 IN_PORT = 2, OUTPUT = 1) ，則 link 如下 ，會多出一條

s1:1 -> s3:1 的 flow

Links (4 total)

	PORT 1	PORT 2	TYPE	DIRECTION
✔	of:0000000000000002/2	of:0000000000000003/1	Direct	A ↔ B
✘	of:0000000000000002/1	of:0000000000000001/1	Direct	A → B
✔	of:0000000000000001/2	of:0000000000000003/2	Direct	A ↔ B
✘	of:0000000000000001/1	of:0000000000000003/1	Direct	A → B

若設計的 s2 flow rule 雙方向 (IN_PORT = 1, OUTPUT = 2) & (IN_PORT = 2, OUTPUT = 1)

s1:1 <-> s3:1 的 flow 可以互相通，並且 s2 的 link 都只有單一方向

Links (4 total)

	PORT 1	PORT 2	TYPE	DIRECTION
✖	of:0000000000000002/2	of:0000000000000003/1	Direct	A → B
✖	of:0000000000000002/1	of:0000000000000001/1	Direct	A → B
✔	of:0000000000000001/2	of:0000000000000003/2	Direct	A ↔ B
✔	of:0000000000000001/1	of:0000000000000003/1	Direct	A ↔ B

Tutorial of Curl

- Create new policy

```
curl -u onos:rocks -X POST -H 'Content-Type: application/json' -d
@flow1.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
```

```
curl -I/ -v/ -s/ (Get http reponse)
```

- Get added policy
  - All policy

```
curl -u onos:rocks -X GET -H 'Accept: application/json'
'http://localhost:8181/onos/v1/flows/of:0000000000000001' | json_pp
```

  - Get added specific flow policy

```
curl -u onos:rocks -X GET -H 'Accept: application/json'
'http://localhost:8181/onos/v1/flows/of:0000000000000001/49539596291667367'
| json_pp
```

- Delete added policy

```
curl -v -u onos:rocks -X DELETE -H 'Accept: application/json'
'http://localhost:8181/onos/v1/flows/of:0000000000000001/49539596291667367'
```

The `port_no` field uniquely identifies a port within a switch. The `hw_addr` field typically is the MAC address for the port; `OFP_ETH_ALEN` is 6. The name field is a null-terminated string containing a human-readable name for the interface. The value of `OFP_MAX_PORT_NAME_LEN` is 16.

The port numbers use the following conventions:

```
/* Port numbering. Ports are numbered starting from 1. */
enum ofp_port_no {
    /* Maximum number of physical and logical switch ports. */
    OFPP_MAX        = 0xffffff00,

    /* Reserved OpenFlow Port (fake output "ports"). */
    OFPP_IN_PORT    = 0xfffffff8,  /* Send the packet out the input port.  This
                                      reserved port must be explicitly used
                                      in order to send back out of the input
                                      port. */
    OFPP_TABLE      = 0xfffffff9,  /* Submit the packet to the first flow table
                                      NB: This destination port can only be
                                      used in packet-out messages. */
    OFPP_NORMAL     = 0xfffffffa,  /* Process with normal L2/L3 switching. */
    OFPP_FLOOD      = 0xfffffffb,  /* All physical ports in VLAN, except input
                                      port and those blocked or link down. */
    OFPP_ALL        = 0xfffffffc,  /* All physical ports except input port. */
    OFPP_CONTROLLER = 0xfffffffd,  /* Send to controller. */
    OFPP_LOCAL      = 0xfffffffe,  /* Local openflow "port". */
    OFPP_ANY        = 0xffffffff   /* Wildcard port used only for flow mod
                                      (delete) and flow stats requests. Selects
                                      all flows regardless of output port
                                      (including flows with no output port). */
};
```

```
chmod +x storm.sh
```