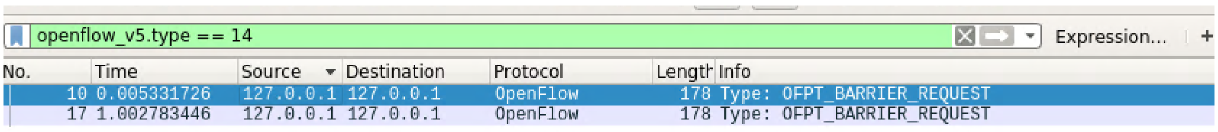
0516094 SDN-NFV Project 2

Part 1

1.

2個



```
openflow_v5.type == 14

No.  Time         Source     Destination Protocol  Length Info
  10 0.005331726  127.0.0.1  127.0.0.1   OpenFlow     178 Type: OFPT_BARRIER_REQUEST
  17 1.002783446  127.0.0.1  127.0.0.1   OpenFlow     178 Type: OFPT_BARRIER_REQUEST
```

2.

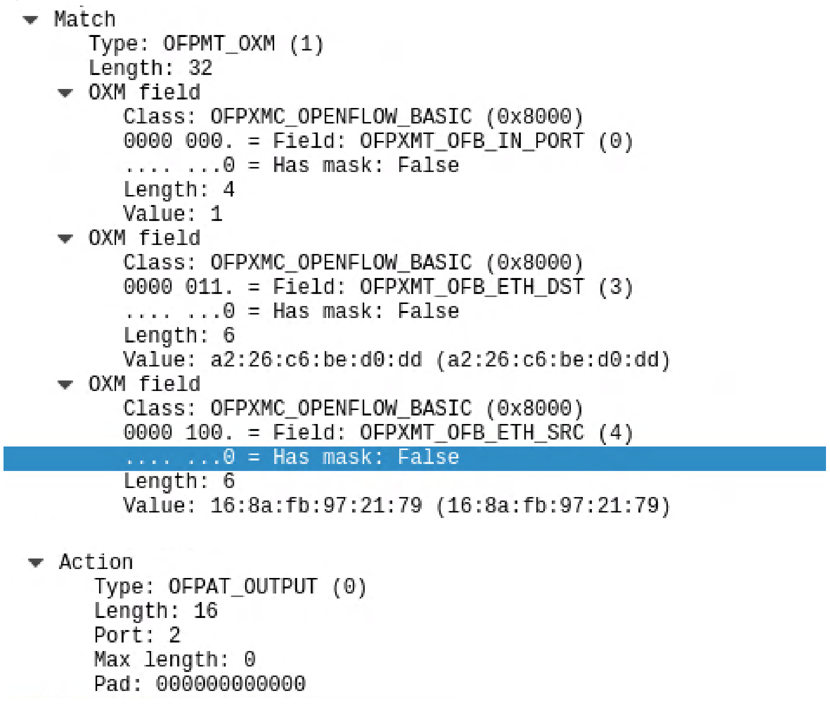
a.

第一個封包：

```
▼ 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: 16:8a:fb:97:21:79 (16:8a:fb:97:21:79)
   ▼ OXM field
        Class: OFPXMC_OPENFLOW_BASIC (0x8000)
        0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
        .... ...0 = Has mask: False
        Length: 6
        Value: a2:26:c6:be:d0:dd (a2:26:c6:be:d0:dd)

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

第二個封包：

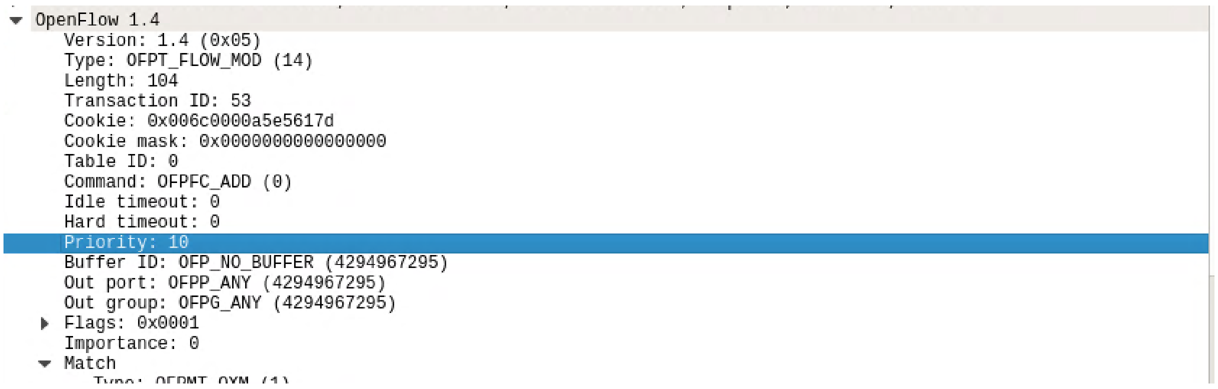


```
▼ Match
    Type: OFPMT_OXM (1)
    Length: 32
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 000. = Field: OFPXMT_OFB_IN_PORT (0)
      .... ...0 = Has mask: False
      Length: 4
      Value: 1
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 011. = Field: OFPXMT_OFB_ETH_DST (3)
      .... ...0 = Has mask: False
      Length: 6
      Value: a2:26:c6:be:d0:dd (a2:26:c6:be:d0:dd)
  ▼ OXM field
      Class: OFPXMC_OPENFLOW_BASIC (0x8000)
      0000 100. = Field: OFPXMT_OFB_ETH_SRC (4)
      .... ...0 = Has mask: False
      Length: 6
      Value: 16:8a:fb:97:21:79 (16:8a:fb:97:21:79)

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

b.

priority為10

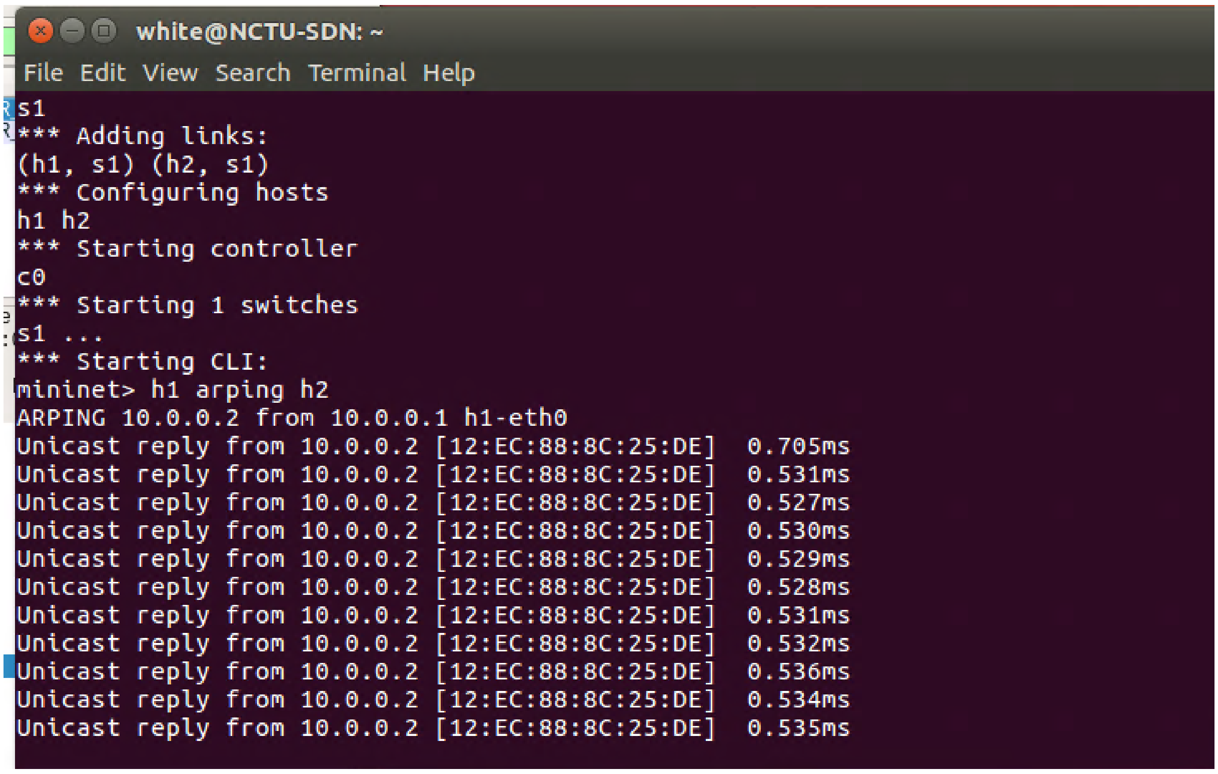


```
▼ OpenFlow 1.4
    Version: 1.4 (0x05)
    Type: OFPT_FLOW_MOD (14)
    Length: 104
    Transaction ID: 53
    Cookie: 0x006c0000a5e5617d
    Cookie mask: 0x0000000000000000
    Table ID: 0
    Command: OFPFC_ADD (0)
    Idle timeout: 0
    Hard timeout: 0
    Priority: 10
    Buffer ID: OFP_NO_BUFFER (4294967295)
    Out port: OFPP_ANY (4294967295)
    Out group: OFPG_ANY (4294967295)
  ▶ Flags: 0x0001
    Importance: 0
  ▼ Match
      Type: OFPMT_OXM (1)
```

Part 2

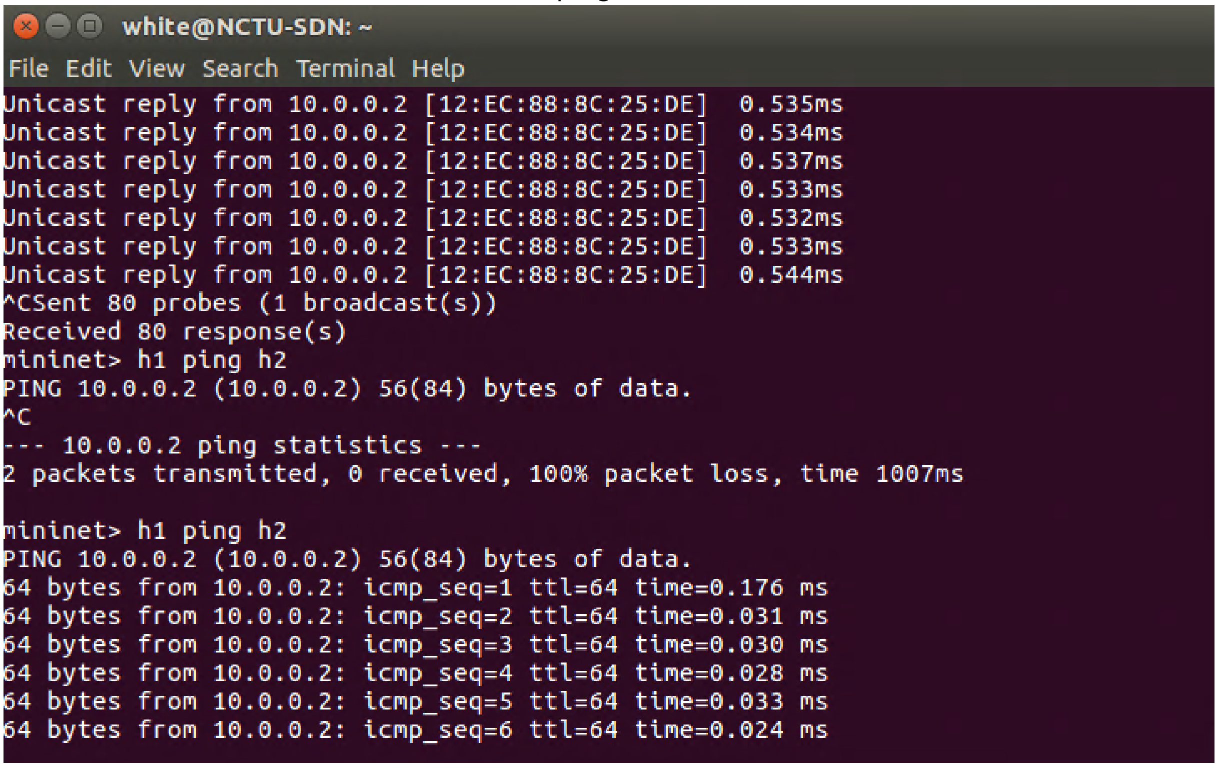
1.

`flows_s1-1_0516094.json` 為當s1接收到來自port為1且是ARP封包時，會將封包從port 2送出去。
`flows_s1-2_0516094.json` 為當s1接收到來自port為2且是ARP封包時，會將封包從port 1送出去。
因此安裝完這兩條flow rules後，h1即可成功arping到h2。



```
s1
*** 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 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.705ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.531ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.527ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.530ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.529ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.528ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.531ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.532ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.536ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.534ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.535ms
```

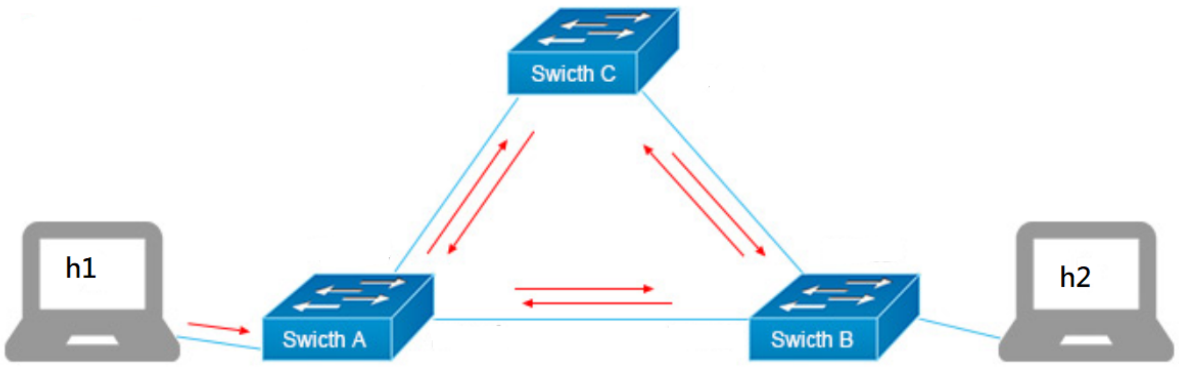
`flows_s1-3_0516094.json` 為當s1接收到來自port為2且是IPv4封包時，會將封包從port 1送出去。
`flows_s1-4_0516094.json` 為當s1接收到來自port為1且是IPv4封包時，會將封包從port 2送出去。
因此安裝完這兩條flow rules後，h1即可成功ping到h2。



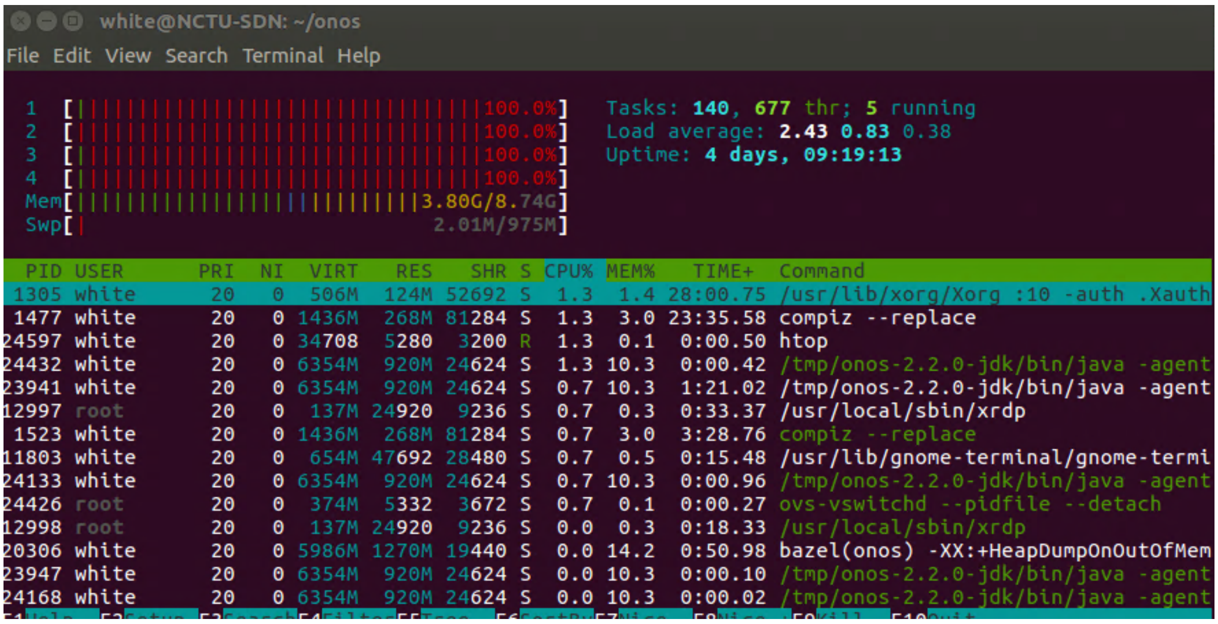
```
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.535ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.534ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.537ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.533ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.532ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.533ms
Unicast reply from 10.0.0.2 [12:EC:88:8C:25:DE]  0.544ms
^CSent 80 probes (1 broadcast(s))
Received 80 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 1007ms

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=0.176 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.030 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.028 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.033 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.024 ms
```

Part 3



我先創造出上面的topology，接著安裝附檔裡的flow rules，並在mininet中執行 `h1 arping h2` ，可以發現CPU使用率為100%。



white@NCTU-SDN: ~/onos

File Edit View Search Terminal Help

```
  1  [||||||||||||||||||||||||||||||||100.0%]   Tasks: 140, 677 thr; 5 running
  2  [||||||||||||||||||||||||||||||||100.0%]   Load average: 2.43 0.83 0.38
  3  [||||||||||||||||||||||||||||||||100.0%]   Uptime: 4 days, 09:19:13
  4  [||||||||||||||||||||||||||||||||100.0%]
  Mem[|||||||||||||||||||||||||||3.80G/8.74G]
  Swp[|                          2.01M/975M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1305 white      20   0  506M  124M 52692 S  1.3  1.4 28:00.75 /usr/lib/xorg/Xorg :10 -auth .Xauth
 1477 white      20   0 1436M  268M 81284 S  1.3  3.0 23:35.58 compiz --replace
24597 white      20   0 34708  5280  3200 R  1.3  0.1  0:00.50 htop
24432 white      20   0 6354M  920M 24624 S  1.3 10.3  0:00.42 /tmp/onos-2.2.0-jdk/bin/java -agent
23941 white      20   0 6354M  920M 24624 S  0.7 10.3  1:21.02 /tmp/onos-2.2.0-jdk/bin/java -agent
12997 root       20   0  137M 24920  9236 S  0.7  0.3  0:33.37 /usr/local/sbin/xrdp
 1523 white      20   0 1436M  268M 81284 S  0.7  3.0  3:28.76 compiz --replace
11803 white      20   0  654M 47692 28480 S  0.7  0.5  0:15.48 /usr/lib/gnome-terminal/gnome-termi
24133 white      20   0 6354M  920M 24624 S  0.7 10.3  0:00.96 /tmp/onos-2.2.0-jdk/bin/java -agent
24426 root       20   0  374M  5332  3672 S  0.7  0.1  0:00.27 ovs-vswitchd --pidfile --detach
12998 root       20   0  137M 24920  9236 S  0.0  0.3  0:18.33 /usr/local/sbin/xrdp
20306 white      20   0 5986M 1270M 19440 S  0.0 14.2  0:50.98 bazel(onos) -XX:+HeapDumpOnOutOfMem
23947 white      20   0 6354M  920M 24624 S  0.0 10.3  0:00.10 /tmp/onos-2.2.0-jdk/bin/java -agent
24168 white      20   0 6354M  920M 24624 S  0.0 10.3  0:00.02 /tmp/onos-2.2.0-jdk/bin/java -agent
```

broadcast storm：因為h1會先送出一個arp broadcast封包，switch A收到從h1來的封包後會broadcast到switch B與switch C，switch B收到從switch A來的封包會broadcast到switch C與h2，而switch C收到封包後會在broadcast到switch A 與 switch B，switch A 與 switch B收到封包後又會在broadcast，一直持續broadcast下去，最終就會導致broadcast storm。