

# 電腦網路實驗實驗報告 < 軟體定義網路 >

姓名：翁佳煌

學號：409430030

## 1. 實驗名稱

Mininet & SDN Controller

## 2. 實驗目的

### 課堂實驗 1 和 2:

使用 OpenFlow 控制器後，當 h1 透過 ping 命令向 h2 發送 ICMP 封包時，封包是否會被傳送到控制器進行處理，並透過控制器將封包廣播到除了發送方之外的所有接口。透過觀察 xterm 中執行的 tcpdump，可以了解此處理過程的細節，並理解 OpenFlow 控制器與傳統集線器的運作差異。實驗 1 會看到相同的 ARP 和對應的 ICMP 封包，實驗 2 會看到三個未應答的 ARP 請求在 tcpdump xterms 中。

### 課堂實驗 3:

熟悉 OpenFlow 協定及其在網路交換機中的應用。在這個實驗中，使用 ~pox/pox/misc 裡面的 of\_tutorial.py 和 Mininet 創建一個網路拓撲，透過控制器設定 OpenFlow 規則，控制網路交換機轉發封包。具體而言，實驗中將透過控制器向交換機發送控制消息，指示交換機根據特定的規則轉發封包，並觀察實驗結果中交換機行為的變化，以了解 OpenFlow 控制器與傳統交換機的不同之處。

## 3. 實驗設備

Linux 作業系統之電腦。

### Download & Install Mininet:

```
$ cd
$ git clone https://github.com/mininet/mininet
$ ./mininet/util/install.sh -a
```

### 安裝 pox controller:

```
$ git clone git://github.com/noxrepo/pox
```

## 4. 實驗步驟

### 課堂實驗 1 和 2:

#### 步驟 1.

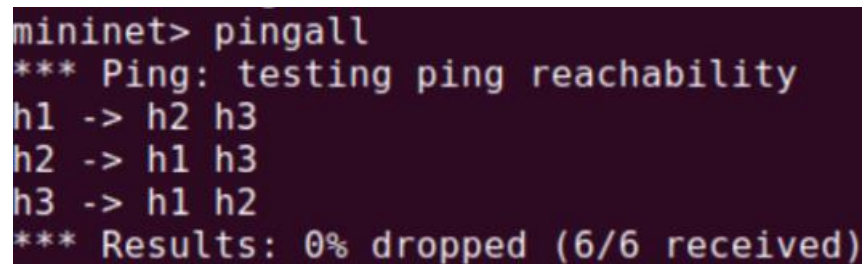
首先開啟第一個 terminal，先 cd 到~/pox 這個路徑裡，然後輸入 ./pox.py log.level --DEBUG misc.of\_tutorial。

#### 步驟 2.

開啟第二個 Terminal，先輸入 sudo mn -c 清除之前的紀錄，以防止未知錯誤發生，接著輸入 sudo mn --topo single,3 --mac --switch ovsk --controller remote。

#### 步驟 3.

在第二個 terminal 輸入 mininet> pingall 確認是否成功連線。

A terminal window with a dark background and light-colored text. The text shows the command 'mininet> pingall' and its output: '\*\*\* Ping: testing ping reachability', followed by connectivity tests for h1, h2, and h3, and finally '\*\*\* Results: 0% dropped (6/6 received)'.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

#### 步驟 4.

接著在第二個 terminal 輸入 create xterm for each host mininet> xterm h1 h2 h3，然後，

In xterm for h2 輸入 tcpdump -X -n -i h2-eth0

In xterm for h3 輸入 tcpdump -X -n -i h3-eth0

In xterm for h1 輸入 ping -c1 10.0.0.2 (實驗 1 和實驗 2 步驟相似只差在實驗 2 在此輸入 ping -c1 10.0.0.5)

## 步驟 5.

課堂實驗 1 觀察結果圖：

```
"Node: h1"
root@lab502:/home/lab502/pox# ping -c1 10.0.0.2
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=3.11 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt_min/avg/max/mdev = 3.109/3.109/3.109/0.000 ms
root@lab502:/home/lab502/pox# ping -c1 10.0.0.5
```

```
"Node: h2"
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  0%()*+,-./0123
0x0050: 3435 3637                                4567
13:52:35.233703 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 25391, seq 1, length
64
0x0000: 4500 0054 0d0b 0000 4001 593c 0a00 0002  E..T....@.Y....
0x0010: 0a00 0001 0000 5074 632f 0001 a399 5464  ....Ptc/....Td
0x0020: 0000 0000 928a 0300 0000 0000 1011 1213  .....l!*
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .....l!*
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  0%()*+,-./0123
0x0050: 3435 3637                                4567
13:52:40.401546 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0001 0800 0604 0001 0000 0000 0002 0a00  .....
0x0010: 0002 0000 0000 0000 0a00 0001  .....
13:52:40.402063 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00  .....
0x0010: 0001 0000 0000 0000 0a00 0002  .....
13:52:40.403088 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000: 0001 0800 0604 0002 0000 0000 0002 0a00  .....
0x0010: 0002 0000 0000 0001 0a00 0001  .....
13:52:40.404063 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0001 0800 0604 0002 0000 0000 0001 0a00  .....
0x0010: 0001 0000 0000 0002 0a00 0002  .....
13:53:51.313549 IP6 fe80::200:ff:fe00:2 > ff02::2: ICMP6, router solicitation, l
ength 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 0200 00ff fe00 0002 ff02 0000 0000 0000  .....
0x0020: 0000 0000 0000 0002 8500 7b2a 0000 0000  .....{*.....
0x0030: 0101 0000 0000 0002  .....
13:54:09.746393 IP6 fe80::200:ff:fe00:1 > ff02::2: ICMP6, router solicitation, l
ength 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 0200 00ff fe00 0001 ff02 0000 0000 0000  .....
0x0020: 0000 0000 0000 0002 8500 7b2c 0000 0000  .....{.....
0x0030: 0101 0000 0000 0001  .....
13:54:24.081571 IP6 fe80::4828:69ff:fe72:bc4d > ff02::2: ICMP6, router solicitation, length 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 0200 00ff fe00 0002 ff02 0000 0000 0000  .....
0x0020: 0000 0000 0000 0002 8500 9f5d 0000 0000  .....H(i..r..M.....
0x0030: 0101 4a28 6972 bc4d  .....
13:54:26.100090 IP6 fe80::4828:69ff:fe72:bc4d,5353 > ff02::fb,5353: 0 [9a] PTR (QM)? _nfs_.tcp.local, PTR (QM)? _ipp_.tc
p.local, PTR (QM)? _ipps_.tcp.local, PTR (QM)? _ftp_.tcp.local, PTR (QM)? _webdav_.tcp.local, PTR (QM)? _webdavs_.tcp.lo
cal, PTR (QM)? _sftp-ssh_.tcp.local, PTR (QM)? _smb_.tcp.local, PTR (QM)? _afpovertcp_.tcp.local. (141)
0x0000: 5000 5b26 0095 11ff fe80 0000 0000 0000  .....%.....
0x0010: 4828 69ff fe72 bc4d ff02 0000 0000 0000  .....H(i..r..M.....
0x0020: 0000 0000 0000 00fb 14e9 14e9 0095 6c0d  .....l.....
```

```
"Node: h3"
0x0020: 0000 0000 928a 0300 0000 0000 1011 1213  .....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .....l!*
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  0%()*+,-./0123
0x0050: 3435 3637                                4567
13:52:35.235195 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 25391, seq 1, length
64
0x0000: 4500 0054 0d0b 0000 4001 593c 0a00 0002  E..T....@.Y....
0x0010: 0a00 0001 0000 5074 632f 0001 a399 5464  ....Ptc/....Td
0x0020: 0000 0000 928a 0300 0000 0000 1011 1213  .....l!*
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .....l!*
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  0%()*+,-./0123
0x0050: 3435 3637                                4567
13:52:40.402886 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0001 0800 0604 0001 0000 0000 0002 0a00  .....
0x0010: 0002 0000 0000 0000 0a00 0001  .....
13:52:40.403031 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00  .....
0x0010: 0001 0000 0000 0000 0a00 0002  .....
13:52:40.404079 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0001 0800 0604 0002 0000 0000 0001 0a00  .....
0x0010: 0001 0000 0000 0002 0a00 0002  .....
13:52:40.445861 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000: 0001 0800 0604 0002 0000 0000 0002 0a00  .....
0x0010: 0002 0000 0000 0001 0a00 0001  .....
13:53:51.315257 IP6 fe80::200:ff:fe00:2 > ff02::2: ICMP6, router solicitation, l
ength 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 0200 00ff fe00 0002 ff02 0000 0000 0000  .....
0x0020: 0000 0000 0000 0002 8500 7b2a 0000 0000  .....{*.....
0x0030: 0101 0000 0000 0002  .....
13:54:03.601564 IP6 fe80::4c99:72ff:feaa:a527 > ff02::2: ICMP6, router solicitat
ion, length 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 4c99 72ff feaa a527 ff02 0000 0000 0000  .....L.r.....
0x0020: 0000 0000 0000 0002 8500 b257 0000 0000  .....M.....
0x0030: 0101 4c99 72aa a527  .....N.r.....
13:54:09.747005 IP6 fe80::200:ff:fe00:1 > ff02::2: ICMP6, router solicitation, l
ength 16
0x0000: 5000 0000 0010 3aff fe80 0000 0000 0000  .....
0x0010: 0200 00ff fe00 0001 ff02 0000 0000 0000  .....
0x0020: 0000 0000 0000 0002 8500 7b2c 0000 0000  .....{.....
0x0030: 0101 0000 0000 0001  .....
13:54:26.020372 IP6 fe80::4c99:72ff:feaa:a527,5353 > ff02::fb,5353: 0 [9a] PTR (
QM)? _nfs_.tcp.local, PTR (QM)? _ipp_.tcp.local, PTR (QM)? _ipps_.tcp.local, PTR
(QM)? _ftp_.tcp.local, PTR (QM)? _webdav_.tcp.local, PTR (QM)? _webdavs_.tcp.lo
cal, PTR (QM)? _sftp-ssh_.tcp.local, PTR (QM)? _smb_.tcp.local, PTR (QM)? _afpov
ertcp_.tcp.local. (141)
0x0000: 5000 5b24 0095 11ff fe80 0000 0000 0000  .....
0x0010: 4c99 72ff feaa a527 ff02 0000 0000 0000  .....L.r.....
0x0020: 0000 0000 0000 00fb 14e9 14e9 0095 6290  .....B.....
0x0030: 0000 0000 0009 0000 0000 0000 045f 6e56  ....._nf
```

## 課堂實驗 2 觀察結果圖：

```
"Node: h1"
root@lab502:/home/lab502/pox# ping -c1 10.0.0.2
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=2.64 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.637/2.637/2.637/0.000 ms
root@lab502:/home/lab502/pox# ping -c1 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.5 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
root@lab502:/home/lab502/pox#
```

```
"Node: h2"

14:01:12.515422 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
14:01:13.522370 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
14:01:14.546599 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
```

```
"Node: h3"

14:01:12.515424 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
14:01:13.522372 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
14:01:14.546602 ARP, Request who-has 10.0.0.5 tell 10.0.0.1, length 28
    0x0000: 0001 0800 0604 0001 0000 0000 0001 0a00 .....
    0x0010: 0001 0000 0000 0000 0a00 0005 .....
```

### 課堂實驗 3:

#### 步驟 1.

開啟第一個 Terminal，並 cd ~/pox/pox/misc，然後輸入 gedit of\_tutorial.py。

#### 步驟 2.

修改相關函式，如下圖 1。

```
def _handle_PacketIn (self, event):
    """
    Handles packet in messages from the switch.
    """
    packet = event.parsed # This is the parsed packet data.
    if not packet.parsed:
        log.warning("Ignoring incomplete packet")
        return

    packet_in = event.ofp # The actual ofp_packet_in message.

    # Comment out the following line and uncomment the one after
    # when starting the exercise.
    #self.act_like_hub(packet, packet_in)
    self.act_like_switch(packet, packet_in)

def act_like_switch (self, packet, packet_in):
    """
    Implement switch-like behavior.
    """
    # DELETE THIS LINE TO START WORKING ON THIS (AND THE ONE BELOW!) #
    # Here's some psuedocode to start you off implementing a learning
    # switch. You'll need to rewrite it as real Python code.
    # Learn the port for the source MAC
    src_mac=packet.src
    dst_mac=packet.dst
    in_port=packet.in_port
    print("in_port:", in_port)
    if src_mac not in self.mac_to_port:
        self.mac_to_port[src_mac]=in_port
        print("mac_to_port:", self.mac_to_port)

    #if the port associated with the destination MAC of the packet is known:
    # Send packet out the associated port
    if dst_mac in self.mac_to_port:
        #write a flow entry
        eth_dst_match=of.ofp_match(dl_dst=dst_mac)
        msg=of.ofp_flow_mod()
        msg.match=eth_dst_match
        msg.priority=1
        print("dst_in_switch_port:", self.mac_to_port[dst_mac])
        msg.actions.append(of.ofp_action_output(port=self.mac_to_port[dst_mac]))
        self.connection.send(msg)
        self.resend_packet(packet_in, self.mac_to_port[dst_mac])
    else:
        # Flood the packet out everything but the input port
        # This part looks familiar, right?
        self.resend_packet(packet_in, of.OFPP_ALL)

    # DELETE THIS LINE TO START WORKING ON THIS #
```

▲圖 1

### 步驟 3.

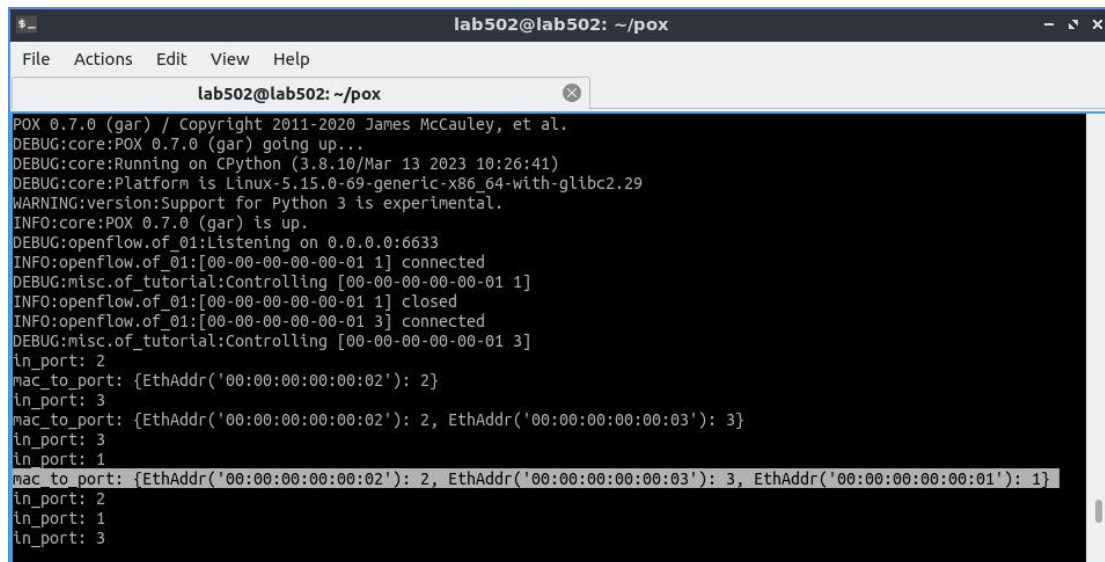
重啟 POX 和 Mininet ,接著輸入:

```
mininet> h1 ping h2
```

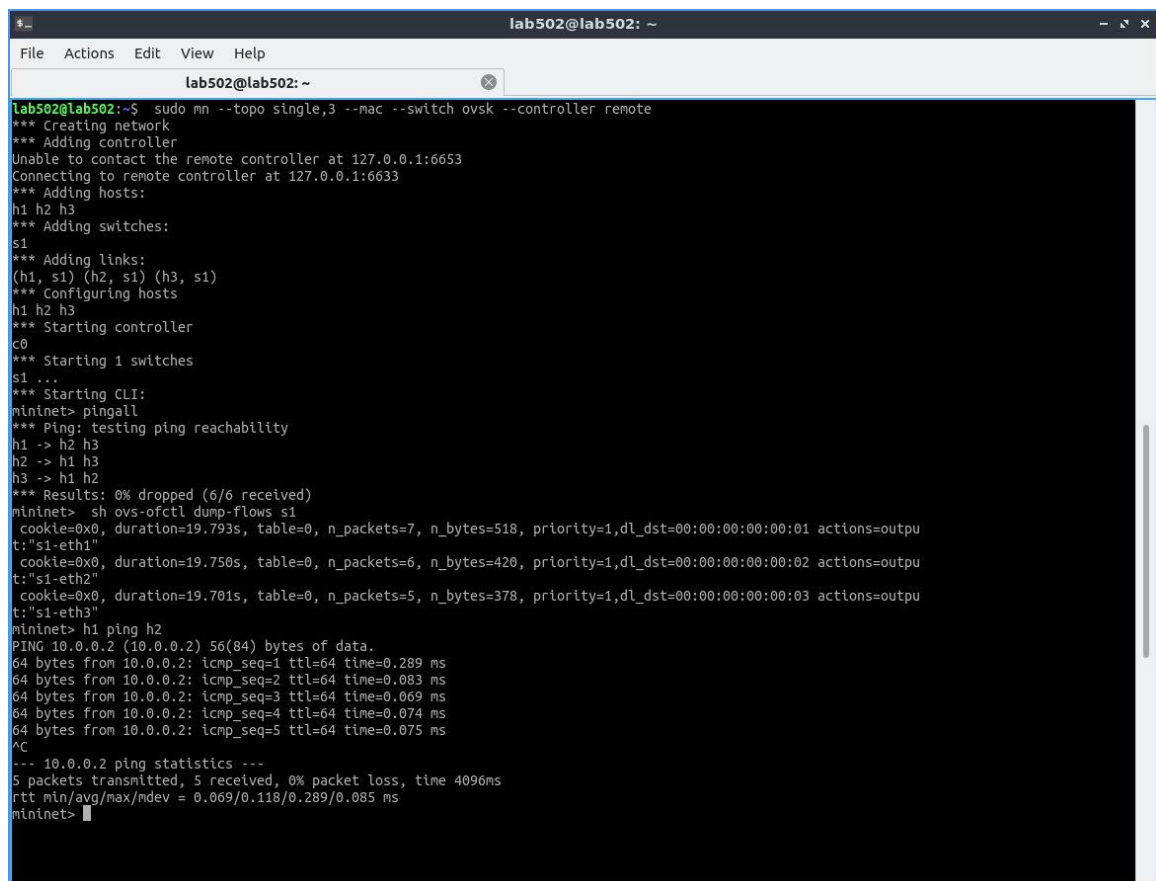
```
mininet> sh ovs-ofctl dump-flows s1
```

### 步驟 4.

觀察結果。



```
lab502@lab502: ~/pox
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.8.10/Mar 13 2023 10:26:41)
DEBUG:core:Platform is Linux-5.15.0-69-generic-x86_64-with-glibc2.29
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 1]
INFO:openflow.of_01:[00-00-00-00-00-01 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 3]
in_port: 2
mac_to_port: {EthAddr('00:00:00:00:00:02'): 2}
in_port: 3
mac_to_port: {EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3}
in_port: 3
in_port: 1
mac_to_port: {EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:01'): 1}
in_port: 2
in_port: 1
in_port: 3
```



```
lab502@lab502:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=19.793s, table=0, n_packets=7, n_bytes=518, priority=1,dl_dst=00:00:00:00:00:01 actions=output
 t:"s1-eth1"
 cookie=0x0, duration=19.750s, table=0, n_packets=6, n_bytes=420, priority=1,dl_dst=00:00:00:00:00:02 actions=output
 t:"s1-eth2"
 cookie=0x0, duration=19.701s, table=0, n_packets=5, n_bytes=378, priority=1,dl_dst=00:00:00:00:00:03 actions=output
 t:"s1-eth3"
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=0.289 ms
 64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.083 ms
 64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.069 ms
 64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.074 ms
 64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.075 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4096ms
rtt min/avg/max/mdev = 0.069/0.118/0.289/0.085 ms
mininet>
```



## 5. 問題與討論

### 課堂實驗 1 和 2:

第一個實驗中，觀察到 ping 數據包被傳送到控制器，然後被控制器洪泛到除了發送端口之外的所有接口。這種行為在一個簡單的網路中可能是可接受的，但在一個大型網路中，可能會導致控制器被大量流量淹沒，從而影響整個網路的性能。因此，這個實驗提醒我們，在實際網路中需要適當的 OpenFlow 規則，以最小化控制器的參與和最大化網路的自主性。

第二個實驗中觀察到的三個未回覆的 ARP 請求是因為在 Mininet 拓撲中只有三個主機 (h1, h2 和 h3)，因此當主機 h1 向 10.0.0.5 發出 ARP 請求時，因為該 IP 地址沒有被分配給任何主機，因此不會有任何主機回應這個 ARP 請求，從而導致三個 ARP 請求都未被回覆。

### 課堂實驗 3:

在 Part A 中，當控制器向交換機發送控制消息時，交換機會根據該消息直接轉發封包，而不需要事先設置 OpenFlow 規則。

在 Part B 中，控制器除了向交換機發送控制消息外，還設置了一條 OpenFlow 規則，用來匹配封包的源 MAC 地址和目的 MAC 地址。當封包匹配到該規則時，交換機會根據規則中的指令轉發封包，而不需要再向控制器請求轉發指示。此外，在 mininet 命令行中可以使用 `ovs-ofctl dump-flows s1` 命令查看交換機的流表規則，以驗證規則是否已經配置成功。

## 6. 心得與感想

這次實驗的重點在於控制器和 OpenFlow 交換機之間的通訊協議，以及如何使用控制器來控制網路流量。透過實驗，我了解到 OpenFlow 控制器使用 OpenFlow 協議來控制網路交換機，並且可以通過控制器來設置交換機上的流規則，以實現特定的網路流量控制。因為實驗環境已經被預先設置好了，所以我不需要自己配置環境。我只需要根據指示來操作，就可以完成整個實驗 1 和 2，實驗 3 需要另外思考如何改 py 程式碼。此外，教材中也提供了詳細的指導和說明，這對我來說非常有幫助。

總體來說，這次的 OpenFlow 實驗讓我對軟體定義網路有了更深入的了解，也讓我學會了如何設置 OpenFlow 網路和控制網路流量。我相信這些知識和技能將對我的未來研究和工作非常有用。

## 7. 參考文獻

<https://github.com/mininet/mininet>

<http://github.com/Hsun111/MininetTopology>

<https://www.vmware.com/tw/topics/glossary/content/software-defined-networking.html>

<https://ithelp.ithome.com.tw/articles/10235434>

<https://www.openedu.tw/course?id=1002>