SDN - Lab4 Report

R03922133 黃子軒

Code design:

首先，我先在 46 行加入一段邏輯，目的是將新接收的 port 存取到 new_inport 變數中，方便之後做使用。

為了要 learn mac address 以避免下次又 flooding 一次，我在 64 行加入了這段邏輯: self.mac_to_port[dpid][src] = new_inport

其中的變數 mac_to_port 是原先定義好用來存放 mac address 的表格，而變數 new_inport 則是剛剛定義新接收到的 port。

接著我要去查詢新封包的目標位址是否已存在 table 中，如果存在，就將輸出設為該 table 的號碼使該封包能送達目的地，否則就進行 flooding，因此 code 如下:

```
if dst in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][dst]
else:
    out_port = ofproto.OFPP_FLOOD
```

接著我們要定義須立即執行的行為，因此加了這段邏輯:

```
actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]
```

OFPActionOutput 為一個 class 包含了 out_port 這個物件，目的是告知 switch 要將 packet 送至哪個 port，而這個 class 需要一個 switch 來呼叫他，因此前面再多加 datapath.ofproto_parser 來代表這個 flow 所協商的 switch。

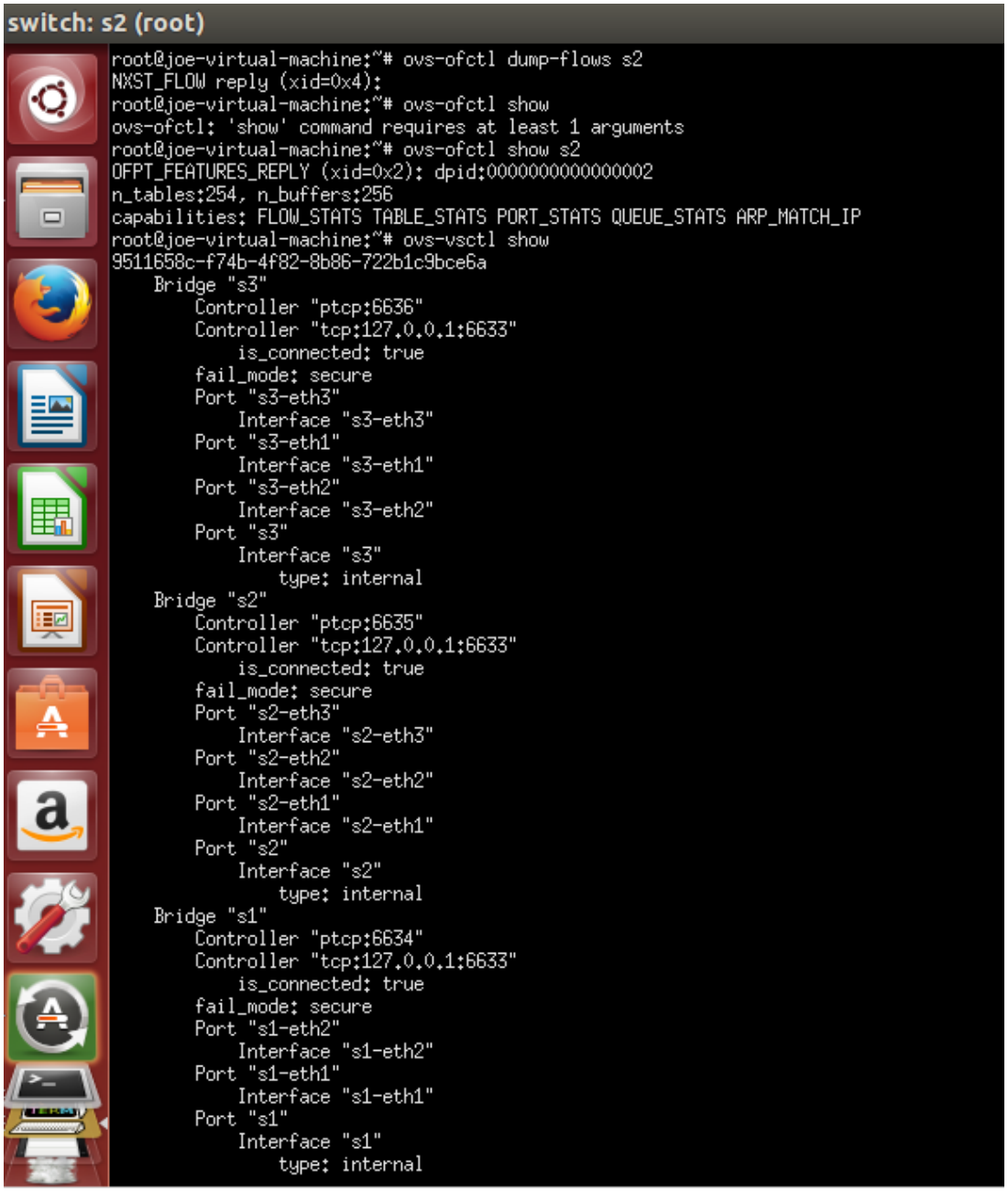
最後，更新完 Mac Table、out_port 和 actions 後，就在交換器的 Flow table 做新增的動作，透過 add_flow 去實作，而特別注意的是只能在非 flooding 的條件下才能做這件事，因此加了這段邏輯:

```
if out_port != ofproto.OFPP_FLOOD:
    self.add_flow(datapath, new_inport, dst, actions)。
```

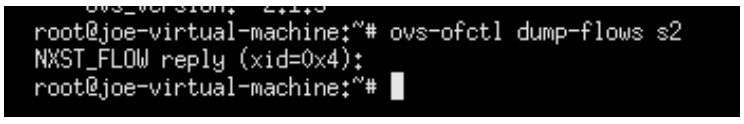
Observations:

首先先利用 mininet 建立一個簡單的小型拓樸網路，形式為一高度三的二元樹，先由一個 controller 連到 Switch1，Switch1 有兩個兒子分別是 Switch2、Switch3，Switch2 和 Swich3 又分別有兩個兒子，各是 h1、h2、h3 和 h4。

先利用 xterm 查看任何一個 switch 的狀態，可清楚看到各個 Switch 所連到的 port 和 interface。



檢查 switch2 的 Flow Table，可發現是乾淨空白的。



接著我們對 mininet 執行 h1 ping h2 的動作，並觀察 switch2 的 flow table，可發現 table 已有被填入內容，內容為我們剛剛所寫的 code 所控制。

```
root@joe-virtual-machine:~# ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=21.658s, table=0, n_packets=11, n_bytes=1022, idle_age=12,
 in_port=2,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=21.639s, table=0, n_packets=10, n_bytes=924, idle_age=12,
in_port=1,dl_dst=00:00:00:00:00:02 actions=output:2
root@joe-virtual-machine:~#
```

觀察到:當 in_port 為 2 時，意即該封包從 h2 而來，而他的 output 會被設為 1，表示以後若有封包從 h2 進來，都會將之送往 h1，反之亦然。

開啟 wireshark 並觀察 h1 ping h2 會有哪些封包透過哪些協定來傳送，觀察後如下:

1. ARP request : h1 並不知道 h2 的 MAC address，因此 ARP 會以廣播方式告知 h2、h3 和 h4 自己的 MAC address。

2. ARP reply : h2 回覆了 h1 的要求。

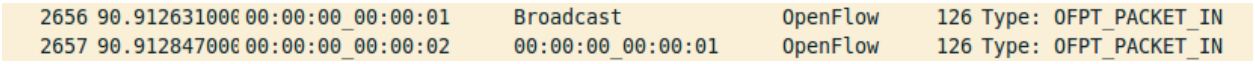
```
2660 90.912157006 00:00:00_00:00:01     Broadcast           ARP          42 Who has 10.0.0.2?  Tell 10.0.0.1
2661 90.912227006 00:00:00_00:00:02     00:00:00_00:00:01   ARP          42 10.0.0.2 is at 00:00:00:00:00:02
2662 90.953648006 00:00:00_00:00:01     Broadcast           ARP          42 Who has 10.0.0.2?  Tell 10.0.0.1
2663 90.903704006 00:00:00_00:00:01     Broadcast           ARP          42 Who has 10.0.0.2?  Tell 10.0.0.1
```

3. ICMP echo request : h1 現在已知 h2 的 MAC address，因此發送 echo request 給 h2。

4. ICMP echo reply : h2 此時也已經知道 h1 的 MAC address，因此發送 echo reply 給 h1

```
  32 10.775595006 10.0.0.1    10.0.0.2    ICMP   98 Echo (ping) request  id=0x22c5, seq=2/512, ttl=64 (reply in 3
  33 10.775640006 10.0.0.2    10.0.0.1    ICMP   98 Echo (ping) reply    id=0x22c5, seq=2/512, ttl=64 (request in
  34 10.775557006 10.0.0.1    10.0.0.2    ICMP   98 Echo (ping) request  id=0x22c5, seq=2/512, ttl=64 (reply in 3
  35 10.775648006 10.0.0.2    10.0.0.1    ICMP   98 Echo (ping) reply    id=0x22c5, seq=2/512, ttl=64 (request in
  36 11.774640006 10.0.0.1    10.0.0.2    ICMP   98 Echo (ping) request  id=0x22c5, seq=3/768, ttl=64 (reply in 3
  37 11.774714006 10.0.0.2    10.0.0.1    ICMP   98 Echo (ping) reply    id=0x22c5, seq=3/768, ttl=64 (request in
```

針對 OpenFlow 協定來討論 ： 第一個 Packet_in 的訊息，是因為由 h1 發送的 ARP request 是廣播的方式。 而第二個 Packet_in 訊息則是 h2 回覆給 h1 的 ARP reply，目的地是 h1 的 MAC address，此時 Flow Entry 被新增，而後 h1 向 h2 發送的 ICMP request 又會再一次新增 Flow Entry，當 h2 回覆 ICMP 給 h1 時，會在 Flow table 找到路徑，因此直接將封包送至 h1。



Reference:

http://osrg.github.io/ryu-book/en/html/switching_hub.html
http://osrg.github.io/ryu-book/zh_tw/Ryubook.pdf