

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

OpenFlow 協定及基本操作：

1. 啟用環境：sudo python3 ./Homework.py
2. 編寫路由規則 (Flow entries)
3. 於 Mininet 使用 xterm h1 打開 Host 之終端機
4. 在 h1 的終端機上執行指令 bash ./run_test.sh 進行測試，程式過程及結果會寫在 log 上。
5. 測試後把 50001-s.log, 50002-s.log, 50003-s.log 之檔案內容擷圖。

2. 實驗目的

了解 OpenFlow 協定的基本概念和原理，學習如何在環境中啟用 OpenFlow 並執行相關的操作，以及學習如何編寫路由規則，並在 Mininet 模擬網路環境並在虛擬機中執行指令，進而測試並記錄過程含結果，訓練我們分析和評估 OpenFlow 協定的性能和效果。

3. 實驗設備

Linux 作業系統之電腦。

Mininet。

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

4. 實驗步驟

1.

首先些必須安裝相關的套件，如以下指令。

Install from Mininet:

```
sudo apt install wireshark xterm ifconfig mininet
```

Install from APT:

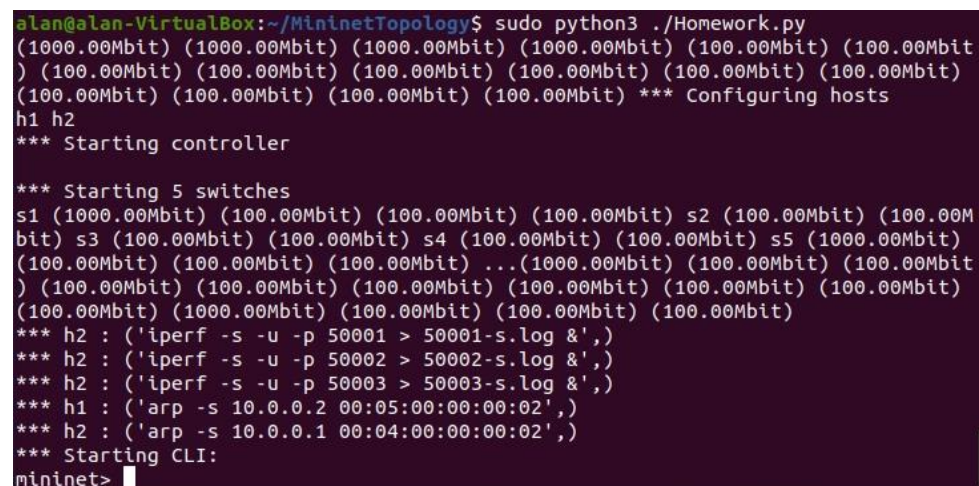
```
sudo apt-get install openvswitch-switch
```

Download the testbed:

```
git clone http://github.com/Hsunlll/MininetTopology
```

2.

開啟 Terminal，並 cd 到 MininetTopology 的資料夾中，然後輸入 `sudo python3 ./Homework.py`，如下圖。



```
alan@alan-VirtualBox:~/MininetTopology$ sudo python3 ./Homework.py
(1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (1000.00Mbit) (100.00Mbit) (100.00Mbit)
) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit)
(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) *** Configuring hosts
h1 h2
*** Starting controller
*** Starting 5 switches
s1 (1000.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) s2 (100.00Mbit) (100.00M
bit) s3 (100.00Mbit) (100.00Mbit) s4 (100.00Mbit) (100.00Mbit) s5 (1000.00Mbit)
(100.00Mbit) (100.00Mbit) (100.00Mbit) ...(1000.00Mbit) (100.00Mbit) (100.00Mbit
) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit)
(100.00Mbit) (1000.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit)
*** h2 : ('iperf -s -u -p 50001 > 50001-s.log &',)
*** h2 : ('iperf -s -u -p 50002 > 50002-s.log &',)
*** h2 : ('iperf -s -u -p 50003 > 50003-s.log &',)
*** h1 : ('arp -s 10.0.0.2 00:05:00:00:00:02',)
*** h2 : ('arp -s 10.0.0.1 00:04:00:00:00:02',)
*** Starting CLI:
mininet>
```

▲圖 1

3.

根據下圖 2，使用 `sudo ovs-ofctl dump-flows <bridge_id>` 寫入每一台 switch 的規則。

舉 s1 的紅色路線為例子，需要輸入：

```
sh ovs-ofctl add-flow s1 dl_type=0x800,nw_proto=17,nw_dst=10.0.0.2,tp_dst=50001,action=output:1
```

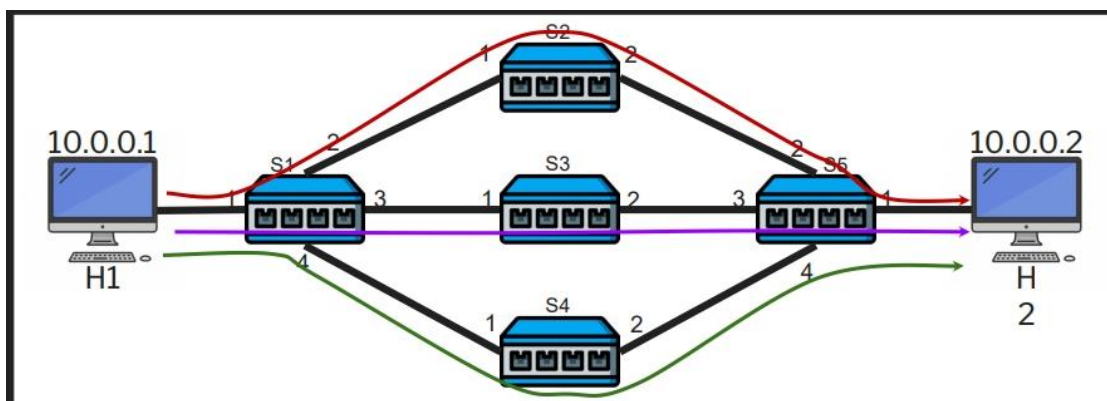
dl_type=0x800 為表示在 OpenFlow 的路由規則 (Flow entry) 中進行 IPv4 封包匹配。

nw_proto=17 表示封包的協議層協議，17 對應於 UDP (User Datagram Protocol) 協議。

tp_dst 則是要根據下圖 3 的顏色路線填入。

"action=output:1" 表示在 OpenFlow 的路由規則 (Flow entry) 中設定了一個動作 (action)，將封包輸出 (output) 到指定的端口。

熟知指令規定後，剩下的步驟皆相似，這裡不多加贅述。



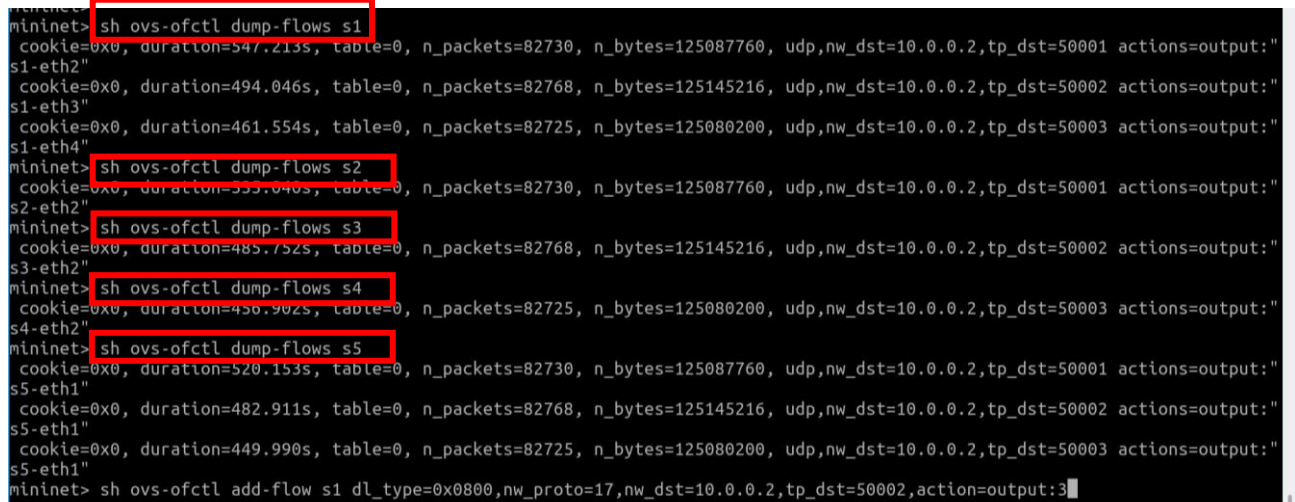
▲圖 2

紅色：UDP Port ID 50001
藍色：UDP Port ID 50002
綠色：UDP Port ID 50003

▲圖 3

4.

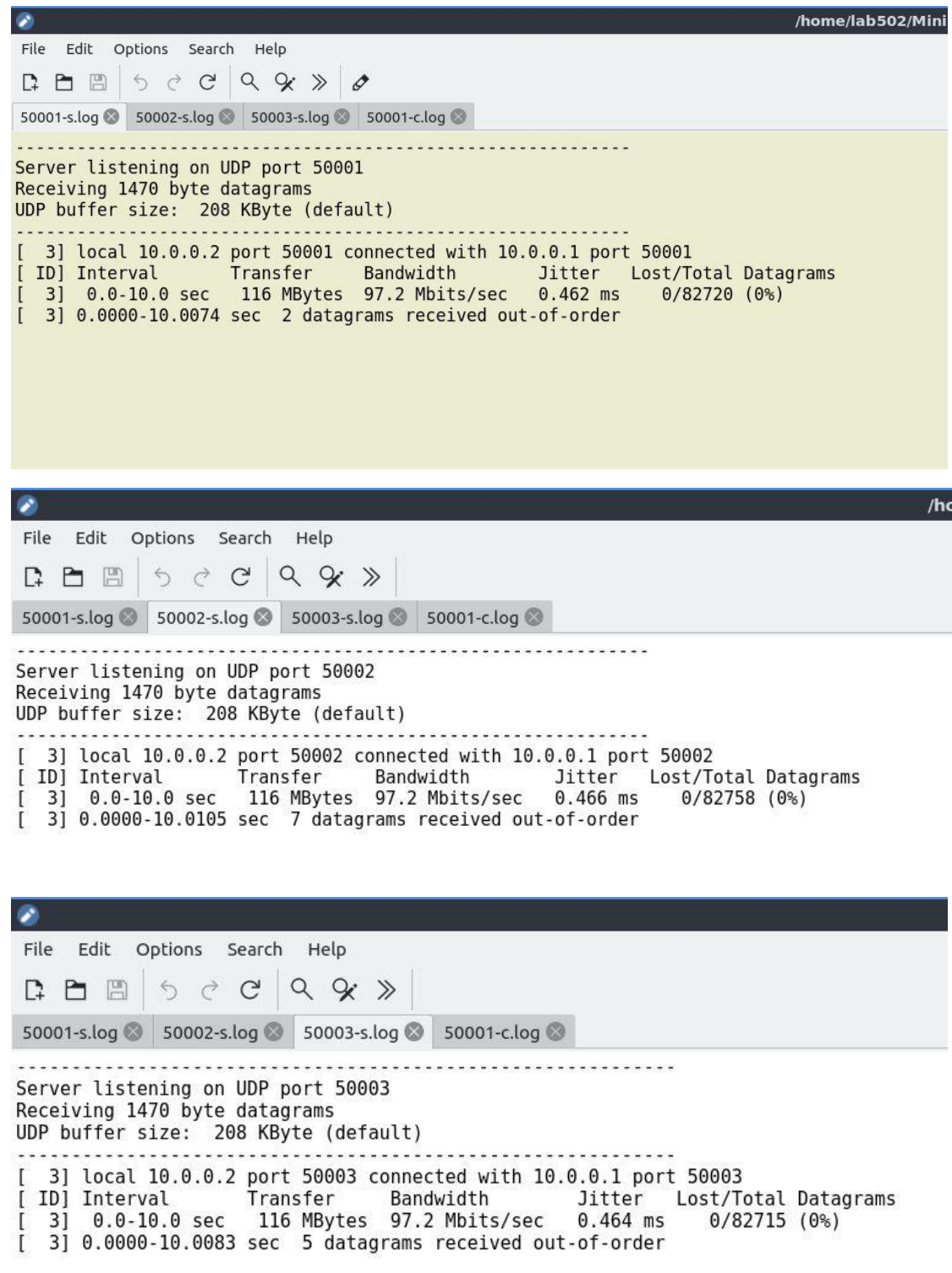
每一台交換機上的規則印出並擷圖：



```
mininet> sh ovs-ofctl dump-flows s1
cookie=0x0, duration=547.213s, table=0, n_packets=82730, n_bytes=125087760, udp,nw_dst=10.0.0.2,tp_dst=50001 actions=output:"
s1-eth2"
cookie=0x0, duration=494.046s, table=0, n_packets=82768, n_bytes=125145216, udp,nw_dst=10.0.0.2,tp_dst=50002 actions=output:"
s1-eth3"
cookie=0x0, duration=461.554s, table=0, n_packets=82725, n_bytes=125080200, udp,nw_dst=10.0.0.2,tp_dst=50003 actions=output:"
s1-eth4"
mininet> sh ovs-ofctl dump-flows s2
cookie=0x0, duration=555.040s, table=0, n_packets=82730, n_bytes=125087760, udp,nw_dst=10.0.0.2,tp_dst=50001 actions=output:"
s2-eth2"
mininet> sh ovs-ofctl dump-flows s3
cookie=0x0, duration=485.752s, table=0, n_packets=82768, n_bytes=125145216, udp,nw_dst=10.0.0.2,tp_dst=50002 actions=output:"
s3-eth2"
mininet> sh ovs-ofctl dump-flows s4
cookie=0x0, duration=456.902s, table=0, n_packets=82725, n_bytes=125080200, udp,nw_dst=10.0.0.2,tp_dst=50003 actions=output:"
s4-eth2"
mininet> sh ovs-ofctl dump-flows s5
cookie=0x0, duration=520.153s, table=0, n_packets=82730, n_bytes=125087760, udp,nw_dst=10.0.0.2,tp_dst=50001 actions=output:"
s5-eth1"
cookie=0x0, duration=482.911s, table=0, n_packets=82768, n_bytes=125145216, udp,nw_dst=10.0.0.2,tp_dst=50002 actions=output:"
s5-eth1"
cookie=0x0, duration=449.990s, table=0, n_packets=82725, n_bytes=125080200, udp,nw_dst=10.0.0.2,tp_dst=50003 actions=output:"
s5-eth1"
mininet> sh ovs-ofctl add-flow s1 dl_type=0x0800,nw_proto=17,nw_dst=10.0.0.2,tp_dst=50002,action=output:3
```

▲圖 4

50001-s.log, 50002-s.log, 50003-s.log 之檔案內容擷圖：



```
-----
Server listening on UDP port 50001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.0.2 port 50001 connected with 10.0.0.1 port 50001
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Totals Datagrams
[ 3]  0.0-10.0 sec   116 MBytes  97.2 Mbits/sec  0.462 ms   0/82720 (0%)
[ 3]  0.0000-10.0074 sec 2 datagrams received out-of-order

-----
Server listening on UDP port 50002
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.0.2 port 50002 connected with 10.0.0.1 port 50002
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Totals Datagrams
[ 3]  0.0-10.0 sec   116 MBytes  97.2 Mbits/sec  0.466 ms   0/82758 (0%)
[ 3]  0.0000-10.0105 sec 7 datagrams received out-of-order

-----
Server listening on UDP port 50003
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 3] local 10.0.0.2 port 50003 connected with 10.0.0.1 port 50003
[ ID] Interval      Transfer    Bandwidth    Jitter    Lost/Totals Datagrams
[ 3]  0.0-10.0 sec   116 MBytes  97.2 Mbits/sec  0.464 ms   0/82715 (0%)
[ 3]  0.0000-10.0083 sec 5 datagrams received out-of-order
```

▲圖 5

5. 問題與討論

路由規則（Flow entries）的設定和操作：在我一開始還沒詢問助教前，漏掉了一些路由規則，導致會跑出錯誤訊息，造成傳送失敗。因此查詢下圖 6 是很重要的步驟和參考工具。

Match Fields	Descriptions
in_port=port	OpenFlow port ID
dl_vlan=vlan	Virtual LAN tag ID
dl_src=xx:xx:xx:xx:xx:xx dl_dst=xx:xx:xx:xx:xx:xx	Ethernet source (or destination) address (MAC addresses)
dl_type=ethertype	Ethernet protocol type
nw_src=ip[/netmask] nw_dst=ip[/netmask]	When dl_type=0x0800 , matches IPv4 source (or destination) address.
arp_spa=ip arp_tpa=ip	When dl_type=0x0806 is specified, matches the arp_spa or arp_tpa field. (IPv4 Addresses)
nw_proto=proto	When dl_type=0x0800 is specified, matches IP protocol type proto, which is specified as a decimal number between 0 and 255
tp_src=port tp_dst=port	When dl_type and nw_proto specify TCP(6) or UD(17), tp_src and tp_dst match the UDP or TCP source or destination port
arp_sha=xx:xx:xx:xx:xx:xx arp_tha=xx:xx:xx:xx:xx:xx	When dl_type=0x0806, arp_sha and arp_tha match the source and target hardware address, respectively

Actions	Descriptions
output:port	Outputs the packet to port
normal	Subjects the packet to the device's normal L2/L3 processing.
flood	Outputs the packet on all switch physical ports other than the port on which it was received and any ports on which flooding is disabled
all	Outputs the packet on all switch physical ports other than the port on which it was received.
in_port	Outputs the packet on the port from which it was received.
drop	Discards the packet, so no further processing or forwarding takes place.
set_field:value->dst	Writes the literal value into the field dst, which should be specified as a name used for matching. Example: set_field:fe80:0123:4567:890a:a6ba:dbff:fefe:59fa->ipv6_src

▲圖 6

測試和驗證實驗結果：可以討論在實驗中執行的測試和驗證步驟，包括執行測試腳本、檢查和分析日誌（log）檔案的內容，並討論測試結果是否符合預期的預期結果。

應用場景和實際應用：可以討論 OpenFlow 協定在實際網絡中的應用場景，例如在軟體定義網絡（SDN）中的應用、網絡流量控制和管理、網絡安全等方面的潛在應用，並討論 OpenFlow 在實際網絡中的優點和限制。

性能和擴展性：OpenFlow 協定在大規模網絡中的性能和擴展性問題，包括控制器的選擇、路由規則的管理和維護、封包轉發的效能等方面的考慮。

安全性：OpenFlow 協定在安全性方面的考慮，包括控制器和交換機之間的通訊安全、路由規則的安全配置、對抗網絡攻擊和威脅的措施等。

未來發展和趨勢：OpenFlow 協定的未來發展和趨勢，包括新的功能和特性、與其他網絡技術的整合、在新興應用和行業中的應用等。

6. 心得與感想

這次的實驗讓我理解到 OpenFlow 協定作為一種軟體定義網絡（SDN）技術，具有在網絡管理和控制方面的優勢。實驗結果顯示，OpenFlow 協定能夠實現動態的流量控制和網絡管理，並能夠根據應用需求進行自動調整，從而提供更靈活和高效的網絡管理方式。

當然，透過這次實作也讓我理解到 OpenFlow 協定具有廣泛的應用潛力，如雲計算、物聯網、5G 網絡、車聯網等。這顯示了 OpenFlow 協定在不斷變化的網絡環境中有著廣泛的應用前景，並能夠應對不同應用場景的需求。

總而言之，OpenFlow 協定作為一種 SDN 技術，具有許多優勢和應用潛力，但也面臨著性能、擴展性和安全性等挑戰。在實際應用中，需要仔細考慮 OpenFlow 協定的適用性和運用方式。

7. 參考文獻

<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>