SDN Experiment 2 Report

Lab 1 環境設定步驟

以下實驗皆在 VMware Workstation 上執行

Open vSwitch(ovs)

建立一台有 Ubuntu 16.04 Desktop 的虛擬機，搭配 4G RAM、8 processers，除了 ovs-vswitchd 不使用 --detach 選項來觀察 log 之外，按照 experiment 2 的說明一步一步安裝 ovs。

Ryu Controller

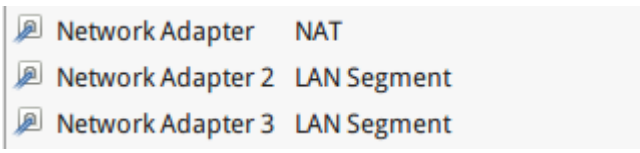
建立一台有 Ubuntu 16.04 Desktop 的虛擬機，搭配 4G RAM、8 processers，同樣跟著 experiment 2 的說明一步一步安裝 Ryu，其中運行 Ryu web UI 的指令有兩個，使用的是 `ryu-manager --observe-links ryu/app/gui_topology/gui_topology.py ryu/app/simple_switch_websocket_13.py` (在 clone 下來的 ryu 資料夾下呼叫)

Hosts

建立兩台有 Ubuntu 16.04 Desktop 的虛擬機，搭配 1G RAM、8 processers

Topology

ovs 的網卡配置為



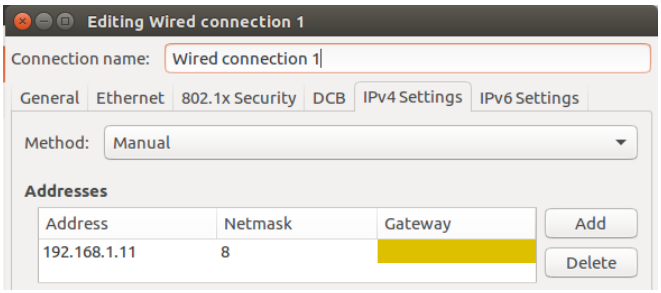
```
ens33     Link encap:Ethernet  HWaddr 00:0c:29:02:90:94
          inet addr:192.168.230.133  Bcast:192.168.230.255  Mask:255.255.255.0
          inet6 addr: fe80::8e61:9b09:dde5:ffeb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:244872 errors:0 dropped:0 overruns:0 frame:0
          TX packets:136165 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:194460613 (194.4 MB)  TX bytes:10156723 (10.1 MB)

ens37     Link encap:Ethernet  HWaddr 00:0c:29:02:90:9e
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:163 errors:0 dropped:0 overruns:0 frame:0
          TX packets:119533 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13556 (13.5 KB)  TX bytes:20946613 (20.9 MB)

ens38     Link encap:Ethernet  HWaddr 00:0c:29:02:90:a8
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:189 errors:0 dropped:0 overruns:0 frame:0
          TX packets:119463 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19281 (19.2 KB)  TX bytes:20934086 (20.9 MB)
```

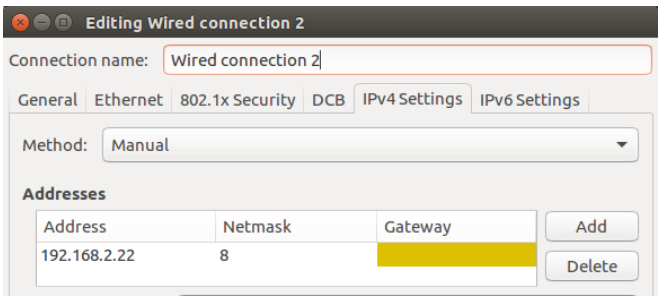
Host 1 & Host 2 透過的就是 LAN segment 接到 ovs 上，，Host 1 接 segment 1， Host 2 接 segment 2。

ryu 的網卡只有一張 NAT，拿到的 IP 為 192.168.230.129 （要讓 ovs 知道）

```
ens33     Link encap:Ethernet  HWaddr 00:0c:29:35:4c:47
          inet addr:192.168.230.129  Bcast:192.168.230.255  Mask:255.255.255.0
```

透過 Ubuntu 的 網路設定介面設定 Host 1 與 Host 2 的固定 IP （作用相當於 `route add` ）



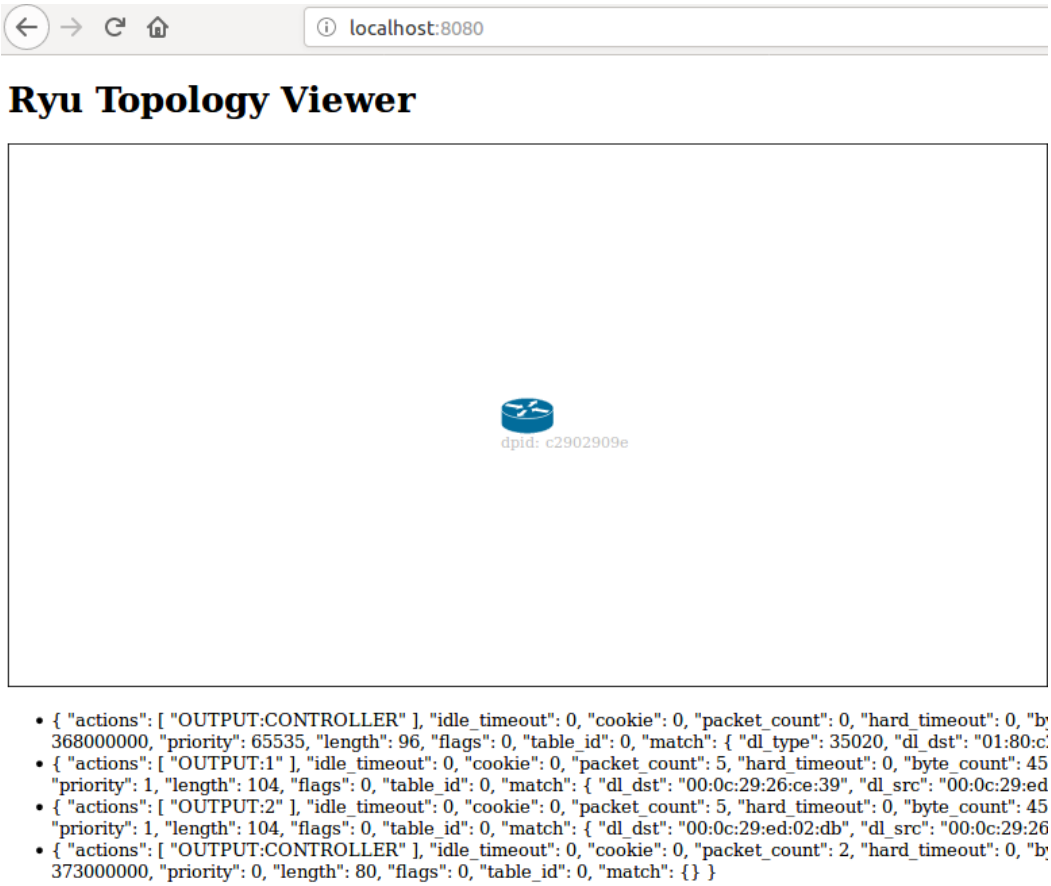


ovs 則透過 `ovs-vsctl` 系列指令設定：

```
sudo ovs-vsctl add-br ovs-br0
sudo ovs-vsctl add-port ovs-br0 ens37
sudo ovs-vsctl add-port ovs-br0 ens38
sudo ovs-vsctl set-controller ovss-br0 tcp:192.168.230.129:6633
```

```
anntsai@ubuntu:~/repos/ovs$ sudo ovs-vsctl show
[sudo] password for anntsai:
33e31f2b-ce90-4a87-ba73-8e8f4340a71f
    Bridge "ovs-br0"
        Controller "tcp:192.168.230.129:6633"
            is_connected: true
        Port "ens37"
            Interface "ens37"
        Port "ens38"
            Interface "ens38"
        Port "ovs-br0"
            Interface "ovs-br0"
                type: internal
```

結果截圖



host 1 ping host 2

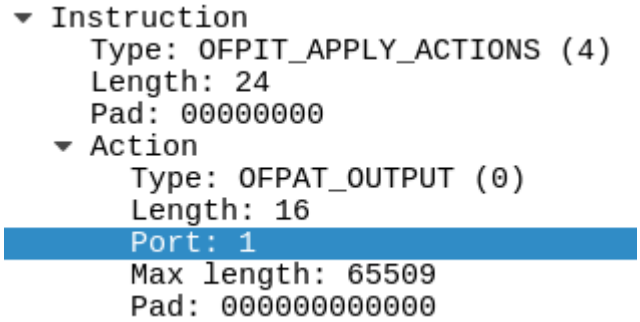
```
anntsai@ubuntu:~$ ping 192.168.2.22
PING 192.168.2.22 (192.168.2.22) 56(84) bytes of data.
64 bytes from 192.168.2.22: icmp_seq=1 ttl=64 time=11.9 ms
64 bytes from 192.168.2.22: icmp_seq=2 ttl=64 time=4.25 ms
64 bytes from 192.168.2.22: icmp_seq=3 ttl=64 time=0.533 ms
64 bytes from 192.168.2.22: icmp_seq=4 ttl=64 time=0.861 ms
```

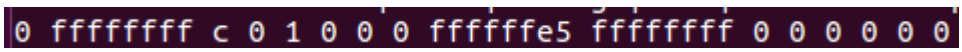
host 2 ping host 1

```
anntsai@ubuntu:~$ ping 192.168.1.11
PING 192.168.1.11 (192.168.1.11) 56(84) bytes of data.
64 bytes from 192.168.1.11: icmp_seq=1 ttl=64 time=4.15 ms
64 bytes from 192.168.1.11: icmp_seq=2 ttl=64 time=1.04 ms
64 bytes from 192.168.1.11: icmp_seq=3 ttl=64 time=1.89 ms
64 bytes from 192.168.1.11: icmp_seq=4 ttl=64 time=1.62 ms
```

Lab 2 如何達成目標

根據要求，要修改 `flow_mod` function 來阻止 ovs 插入 flow entries 到 controller ，首先嘗試在 clone 下來的 ovs 資料夾中尋找有 "flow_mod(" （括號是因為是 function）關鍵字的檔案，綜合檔名、路徑、function 名字和 function 內容覺得最為相關的 function 是 ofproto/ofproto.c 裡面的 `handle_flow_mod` ，接下來從 openflow 下手，找到 `handle_openflow__` 判斷 flow_mod 的部份，找到當 type 為 `OFPTYPE_FLOW_MOD` ，處理的 function 就是 `handle_flow_mod` ，雙重確認後，繼續觀察 `handle_flow_mod` function 的細部實作，第一個內部 function 是 `ofputil_decode_flow_mod` ，可以發現裡面有個型態為 `struct ofpbuf` 的指標變數 ofpacts，可以存取下 table entry 的 elements，如 `data` 跟 `size` ，將他們印出後，然後對應到 Wireshark 的封包：rule 中的 port 跟 max length 是最明顯的對應（如下圖），rule 是在這邊做設定





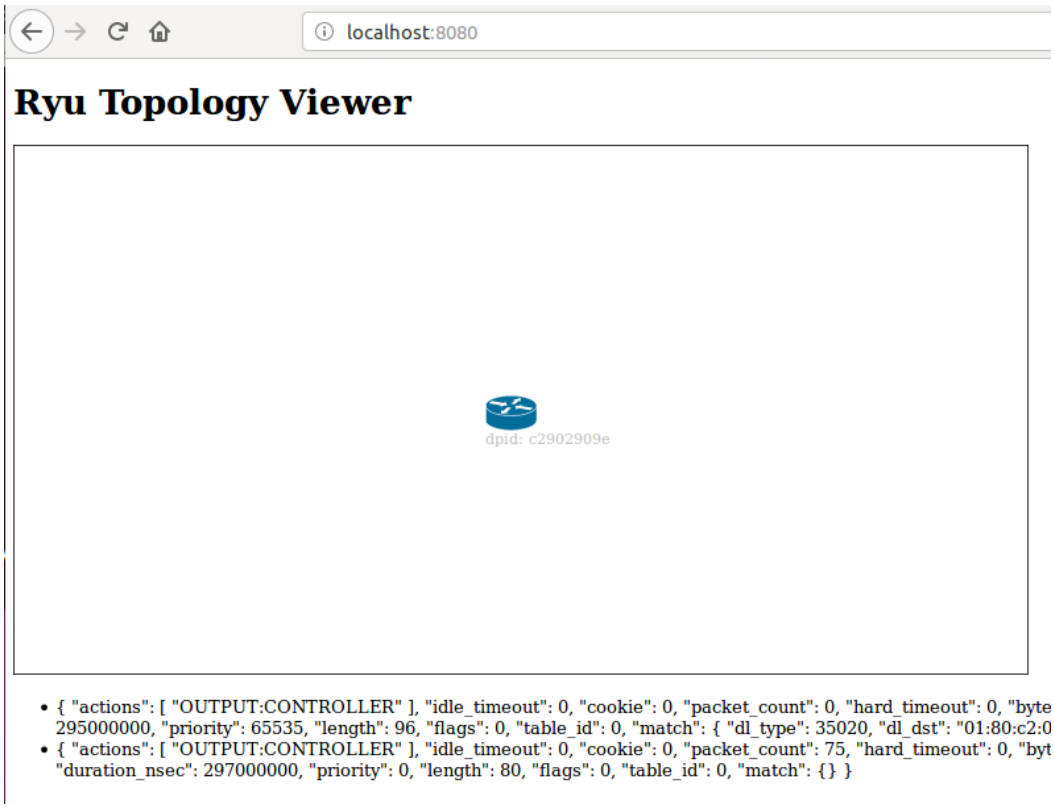
之後在 `handle_flow_mod` 裡的 `handle_flow_mod__` 的 `run_rule_executes` 會下 rule 到 switch，因此在下 rule 前，先讀取 rule 內容，若 rule 中要將封包導向的 port 不是 default port （controller），就不讓 rule 成功下達，controller 也不會收到。

這個部分位於 ofproto/ofproto.c 的 5312 行 到 5304 行

```
//只有在 port 為 65533 (fffd) 時， 會執行 run_rule_executes 的 handle_flow_mod__ 可以執行
if ((unsigned) *(trans_data+4) == 0xfffffffd) {
   error = handle_flow_mod__(ofproto, &ofm, &req);
}
```

如此一來就能防止 ovs 插入 flow entries 到 switch 的 table。

結果截圖



Host 1 ping Host 2

```
anntsai@ubuntu:~$ ping 192.168.2.22
PING 192.168.2.22 (192.168.2.22) 56(84) bytes of data.
From 192.168.1.11 icmp_seq=10 Destination Host Unreachable
From 192.168.1.11 icmp_seq=11 Destination Host Unreachable
From 192.168.1.11 icmp_seq=12 Destination Host Unreachable
From 192.168.1.11 icmp_seq=13 Destination Host Unreachable
```

Host 2 ping Host 1

```
anntsai@ubuntu:~$ ping 192.168.1.11
PING 192.168.1.11 (192.168.1.11) 56(84) bytes of data.
From 192.168.2.22 icmp_seq=9 Destination Host Unreachable
From 192.168.2.22 icmp_seq=10 Destination Host Unreachable
From 192.168.2.22 icmp_seq=11 Destination Host Unreachable
From 192.168.2.22 icmp_seq=12 Destination Host Unreachable
```

Wireshark 會呈現多個 packet in -> packet out -> packet in -> packet flow mod，同作業文件。