計網概 Lab2 Report

●Execution

Part1.Run Mininet and Ryu controller

1. **Steps for running mininet and Ryu controller to ping successfully from host o host.**

   Task1 : Environment Setup

   1. 登入 root
   2. 把資料夾從 github clone 下來

   Task2 : Example of Ryu SDN

   1. 開兩個 terminal 並進入 lab2-jennysun0830/src/ 路徑
      `cd lab2-jennysun0830/src/`
   2. 第一個 terminal 用 Mininet 執行 topo.py
      `sudo mn --custom topo.py --topo topo --link tc --controller remote`
      *如果出現 File exists 則輸入 sudo mn -c 清除先前資料
   3. 第二個 terminal 用 ryu-manager 執行 SImpleController.py
      `sudo ryu-manager SimpleController.py --observe-links`
   4. 先離開 topo.py
      `mininet> exit`
   5. 在離開 SimpleController.py
      `Ctrl-z`
      `mn -c`

   Task3 : Mininet Topology

   1. 根據 topo.png 修改 topo.py 加上 bandwidth, delay, loss rate
   2. 開兩個 terminal 並進入 lab2-jennysun0830/src/ 路徑
      `cd lab2-jennysun0830/src/`
   3. 第一個 terminal 用 Mininet 執行 topo.py
      `sudo mn --custom topo.py --topo topo --link tc --controller remote`
      *如果出現 File exists 則輸入 sudo mn -c 清除先前資料
   4. 第二個 terminal 用 ryu-manager 執行 SImpleController.py
      `sudo ryu-manager SimpleController.py --observe-links`
   5. 測試 h1 到 h2 的連線
      `mininet> h1 ping h2`

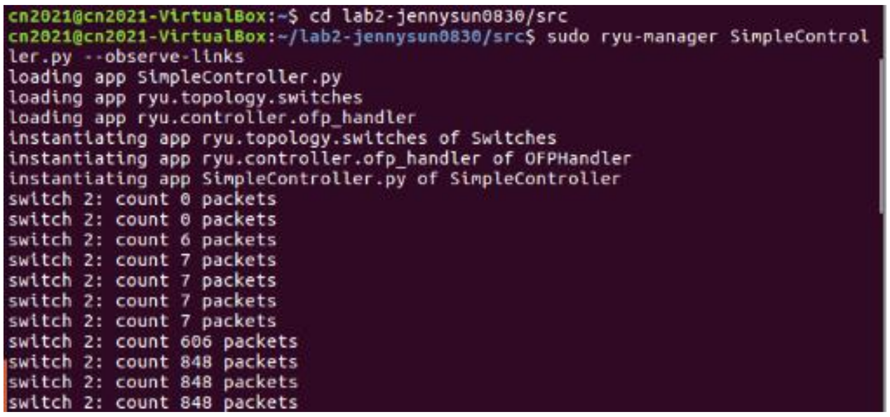
   Task4 : Ryu Controller

   1. 根據圖修改 controller1.py 裡的 switch_feature_handle(self,ev)，重新設

定 forwarding rule
2. Controller2.py 同理

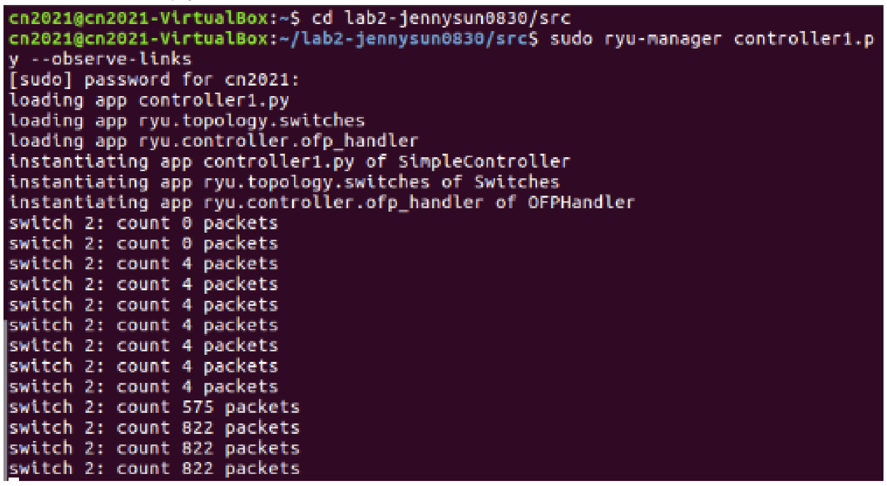
Task5 : Measurement
1. 開兩個 terminal 並進入 lab2-jennysun0830/src/ 路徑
2. 第一個 terminal 用 Mininet 執行 topo.py
3. 第二個 terminal 用 ryu-manager 執行 SImpleController.py / controller1.py / controller2.py

SimpleController.py :



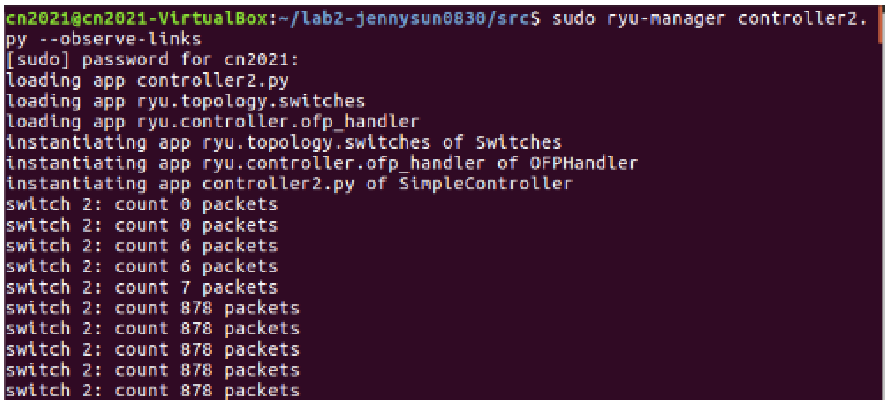
```
cn2021@cn2021-VirtualBox:~$ cd lab2-jennysun0830/src
cn2021@cn2021-VirtualBox:~/lab2-jennysun0830/src$ sudo ryu-manager SimpleControl
ler.py --observe-links
loading app SimpleController.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app SimpleController.py of SimpleController
switch 2: count 0 packets
switch 2: count 0 packets
switch 2: count 6 packets
switch 2: count 7 packets
switch 2: count 7 packets
switch 2: count 7 packets
switch 2: count 7 packets
switch 2: count 606 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
```

Controller1.py :



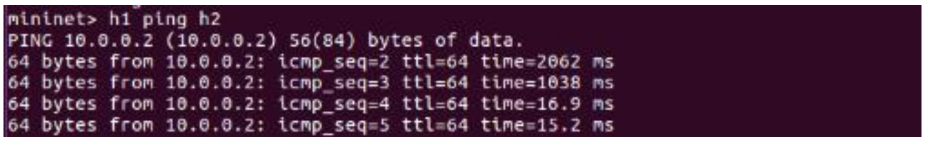
```
cn2021@cn2021-VirtualBox:~$ cd lab2-jennysun0830/src
cn2021@cn2021-VirtualBox:~/lab2-jennysun0830/src$ sudo ryu-manager controller1.p
y --observe-links
[sudo] password for cn2021:
loading app controller1.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller1.py of SimpleController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
switch 2: count 0 packets
switch 2: count 0 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 4 packets
switch 2: count 575 packets
switch 2: count 822 packets
switch 2: count 822 packets
switch 2: count 822 packets
```

Controller2.py



```
cn2021@cn2021-VirtualBox:~/lab2-jennysun0830/src$ sudo ryu-manager controller2.
py --observe-links
[sudo] password for cn2021:
loading app controller2.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app controller2.py of SimpleController
switch 2: count 0 packets
switch 2: count 0 packets
switch 2: count 6 packets
switch 2: count 6 packets
switch 2: count 7 packets
switch 2: count 878 packets
switch 2: count 878 packets
switch 2: count 878 packets
switch 2: count 878 packets
switch 2: count 878 packets
```

4. 測試連線狀態

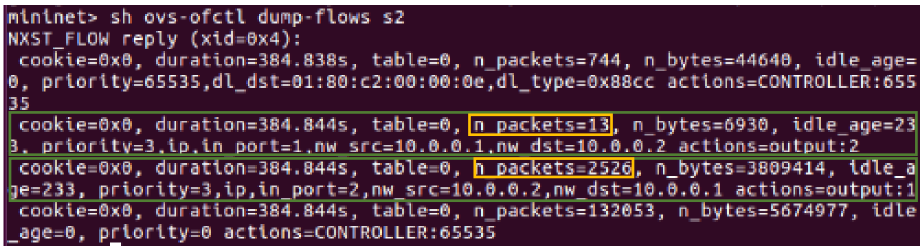


```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2062 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1038 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=16.9 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=15.2 ms
```

5. 測量 bandwidth 並將結果分別存在 Aresult1 / result2 / result3
6. 查看 S2 當前流量表

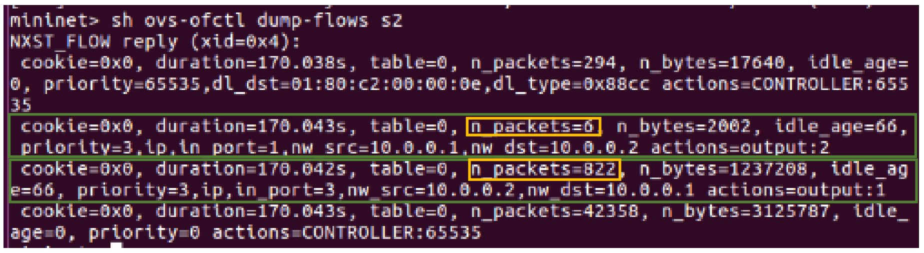
   mininet> sh ovs-ofctl dump-flow s2

   SimpleController.py :



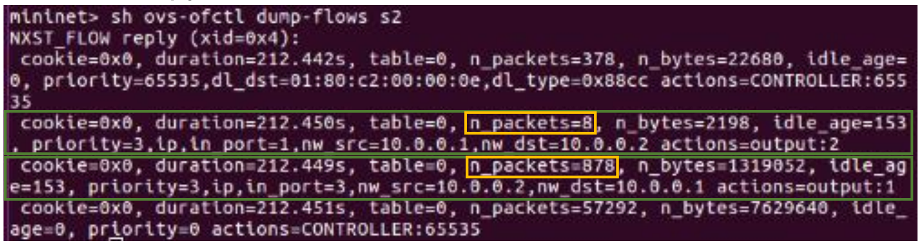
```
mininet> sh ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=384.838s, table=0, n_packets=744, n_bytes=44640, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=384.844s, table=0, n_packets=13, n_bytes=6930, idle_age=23
3, priority=3,ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=384.844s, table=0, n_packets=2526, n_bytes=3809414, idle_a
ge=233, priority=3,ip,in_port=2,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=384.844s, table=0, n_packets=132053, n_bytes=5674977, idle
_age=0, priority=0 actions=CONTROLLER:65535
```

   Controller1.py :



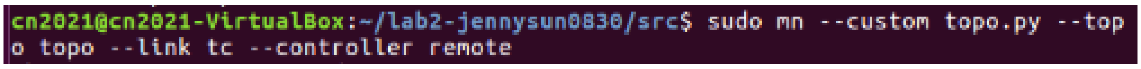
```
mininet> sh ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=170.038s, table=0, n_packets=294, n_bytes=17640, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=170.043s, table=0, n_packets=6, n_bytes=2002, idle_age=66,
 priority=3,ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=170.042s, table=0, n_packets=822, n_bytes=1237208, idle_ag
e=66, priority=3,ip,in_port=3,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=170.043s, table=0, n_packets=42358, n_bytes=3125787, idle_
age=0, priority=0 actions=CONTROLLER:65535
```

   Controller2.py :



```
mininet> sh ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=212.442s, table=0, n_packets=378, n_bytes=22680, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=212.450s, table=0, n_packets=8, n_bytes=2198, idle_age=153
, priority=3,ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=212.449s, table=0, n_packets=878, n_bytes=1319052, idle_ag
e=153, priority=3,ip,in_port=3,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=212.451s, table=0, n_packets=57292, n_bytes=7629640, idle_
age=0, priority=0 actions=CONTROLLER:65535
```

2.What is the meaning of the executing command(both Mininet and Ryu controller)?



```
cn2021@cn2021-VirtualBox:~/lab2-jennysun0830/src$ sudo mn --custom topo.py --top
o topo --link tc --controller remote
```

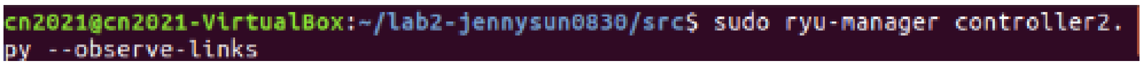
sudo mn :進入 Mininet

--custom topo.py :透過--custom 指定搭配 topo.py 裡的 topo

--topo topo : --topo 後要接 topo.py 最後定義的名稱

--link tc :使用者可用連線進行設定

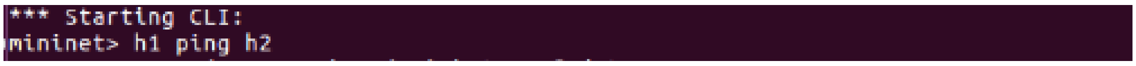
--controller remote :使用 remote controller，即外部 controller 控制



```
cn2021@cn2021-VirtualBox:~/lab2-jennysun0830/src$ sudo ryu-manager controller2.
py --observe-links
```

ryu-manager controller2.py :啟動 Ryu，執行 controller2.py

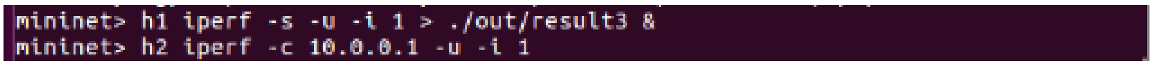
--observe-links :顯示連結間的訊息



```
*** Starting CLI:
mininet> h1 ping h2
```

確認連接狀況

h1 ping h2 :確認 h1-h2 間 SDN 封包傳輸的流程



```
mininet> h1 iperf -s -u -i 1 > ./out/result3 &
mininet> h2 iperf -c 10.0.0.1 -u -i 1
```

測量頻寬

h1 iPerf -s :以 h1 作為 server 端，以 server 模式啟動

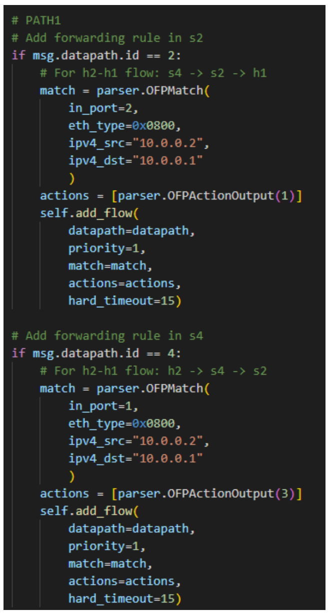
h2 iPerf -c 10.0.0.1 :以 h2 作為 client 端，以 client 模式啟動並連接至 IP address 為 10.0.0.1 的 server 端

-u :使用 UDP

-i 1 :報告時間間隔 1 秒

> ./out/result3 & : 輸出至./out/result3

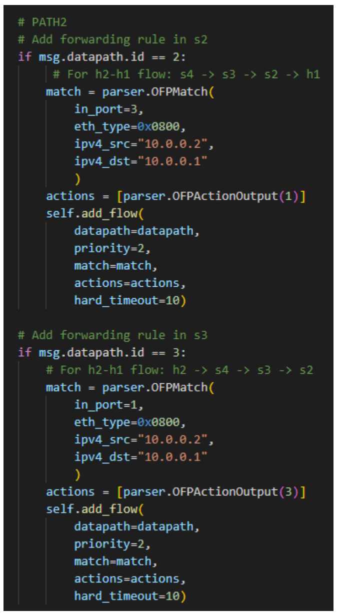
Part2.Handling flow-removed events



```
# PATH1
# Add forwarding rule in s2
if msg.datapath.id == 2:
    # For h2-h1 flow: s4 -> s2 -> h1
    match = parser.OFPMatch(
        in_port=2,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(1)]
    self.add_flow(
        datapath=datapath,
        priority=1,
        match=match,
        actions=actions,
        hard_timeout=15)

# Add forwarding rule in s4
if msg.datapath.id == 4:
    # For h2-h1 flow: h2 -> s4 -> s2
    match = parser.OFPMatch(
        in_port=1,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(3)]
    self.add_flow(
        datapath=datapath,
        priority=1,
        match=match,
        actions=actions,
        hard_timeout=15)
```

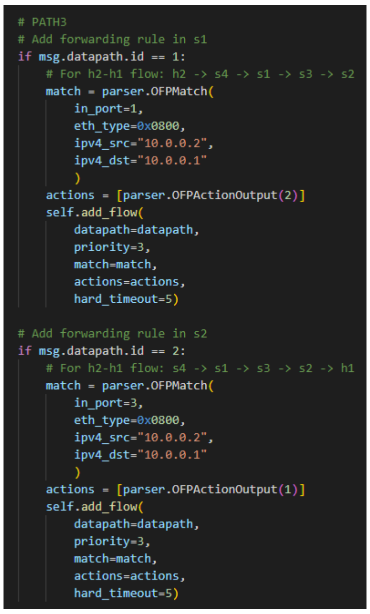
PATH1



```
# PATH2
# Add forwarding rule in s2
if msg.datapath.id == 2:
    # For h2-h1 flow: s4 -> s3 -> s2 -> h1
    match = parser.OFPMatch(
        in_port=3,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(1)]
    self.add_flow(
        datapath=datapath,
        priority=2,
        match=match,
        actions=actions,
        hard_timeout=10)

# Add forwarding rule in s3
if msg.datapath.id == 3:
    # For h2-h1 flow: h2 -> s4 -> s3 -> s2
    match = parser.OFPMatch(
        in_port=1,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(3)]
    self.add_flow(
        datapath=datapath,
        priority=2,
        match=match,
        actions=actions,
        hard_timeout=10)
```

PATH2



```
# PATH3
# Add forwarding rule in s1
if msg.datapath.id == 1:
    # For h2-h1 flow: h2 -> s4 -> s1 -> s3 -> s2
    match = parser.OFPMatch(
        in_port=1,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(2)]
    self.add_flow(
        datapath=datapath,
        priority=3,
        match=match,
        actions=actions,
        hard_timeout=5)

# Add forwarding rule in s2
if msg.datapath.id == 2:
    # For h2-h1 flow: s4 -> s1 -> s3 -> s2 -> h1
    match = parser.OFPMatch(
        in_port=3,
        eth_type=0x0800,
        ipv4_src="10.0.0.2",
        ipv4_dst="10.0.0.1"
        )
    actions = [parser.OFPActionOutput(1)]
    self.add_flow(
        datapath=datapath,
        priority=3,
        match=match,
        actions=actions,
        hard_timeout=5)
```

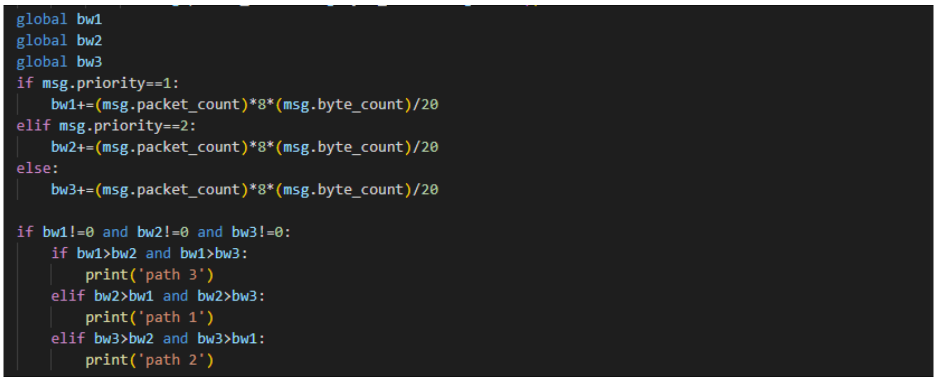
PATH3

設定三個不同的 Path 的 Forwarding rule，設定不同的 priority 即 hard_timeout。Priority 的不同能使我們區分 Flow entry 被執行的優先權，以此區分三條 path。Hard_timeout 分別設定 5、10、15 以此區分三條 path 的運行時間，一旦超過即刪除。

Path1 包含: S2、S4

Path2 包含: S2、S3、S4

Path3 包含: S1、S2、S3、S4


```
global bw1
global bw2
global bw3
if msg.priority==1:
    bw1+=(msg.packet_count)*8*(msg.byte_count)/20
elif msg.priority==2:
    bw2+=(msg.packet_count)*8*(msg.byte_count)/20
else:
    bw3+=(msg.packet_count)*8*(msg.byte_count)/20

if bw1!=0 and bw2!=0 and bw3!=0:
    if bw1>bw2 and bw1>bw3:
        print('path 3')
    elif bw2>bw1 and bw2>bw3:
        print('path 1')
    elif bw3>bw2 and bw3>bw1:
        print('path 2')
```

更動 flow_removed_handlerfunction 的內容

宣告三個全域變數 bw1、bw2、bw3 用來記錄三條路徑使用的頻寬

比較三者，選擇出頻寬使用量最大的代表其收到 ACK 的 packet 最多。

Part3.Problems encountered

跑 ryu-manager 的時候一直跑不出 switch2 count packet，或是 switch2 一直只有 count 到 0 packet，後來發現是 code 裡有錯誤，在重新嘗試後便順利得到合理的結果

•Discussion

1. **Describe the differences between packet-in and packet-out in detail**

   Packet_in:

   接受封包時，轉送到 controller

   Packet_out:

   接受到來自 controller 的封包時，轉送到指定的 port

2. **What is "table-miss" in SDN?**

   Table miss:

   在 Flow table 找符合 rule 的 Flow entry 時找不到對應的 Flow entry

   Table miss 處理方式:

   根據 Flow table 內預設的 rule 動作，可能方法有

   (1) 直接丟棄

   (2) 轉發給後續 Flow table

   (3) 封裝成 packet_in 送往 controller

3. **Why is ("app_manager.RyuApp) adding after the declaration of class in SimpleController.py?**

   因為要時做 Ryu 應用程式必須計成 app_manager.RyuApp，用於加載 Ryu 應用程式，接受從 APP 發送來的訊息，是 base 裡很重要的文件。

4. **What is the meaning of “datapath” in SimpleController.py?**
運用 OpenFlow 的拓樸裡的 switch，OpenFlow 交換器以及 Flowtable 的操作都是透過 Datapath 類別的實體來進行。

5. **Why need to set “eth_type=0x0800” in the flow entry?**
因為須根據 eth_type 填寫產生對應的協定物件，使用 ethernet type 0x0800(IPv4)

6. **Compare the differences between the iPerf results of SimpleController.py , controller1.py and controller2.py. Which forwarding rule is better?Why?**
SimpleController.py :

```
  1.18 MBytes    993 Kbits/sec    0.325 ms    48/   893 (5.4%)
```

Controller1.py :

```
  1.15 MBytes    959 Kbits/sec    0.545 ms    76/   893 (8.5%)
```

Controller2.py :

```
  1.22 MBytes   1.02 Mbits/sec    0.792 ms    22/   893 (2.5%)
```

由數據比較可知三者差別:
在 bandwidth 方面:controller2 > SimpleController > controller1
在 loss 方面:controller2 < SimpleController < controller2

由此可知，controller2.py 的 forwarding rule 是最好的，因為 controller2 在相同時間能傳輸的資料量較大且 loss 較少