109550139 楊竺耘

Execution

Part1:Run Mininet and Ryu controller

1. **Steps for running mininet and Ryu controller to ping successfully from host to host:** First,we need to run topo.py to build the topology and create network via mininet;then,we run the Ryu controller with forwarding rule and use ping command to confirm that packets may reach the destination.
2. **What is the meaning of the executing command:**

   1.--custom: read custom classes from .py file

   2.--topo: set the topology type(linear, torus, tree, single, reversed, mini-mal)

   3.--link: build the link or forbid the link

   4.--controller: set the type of controller

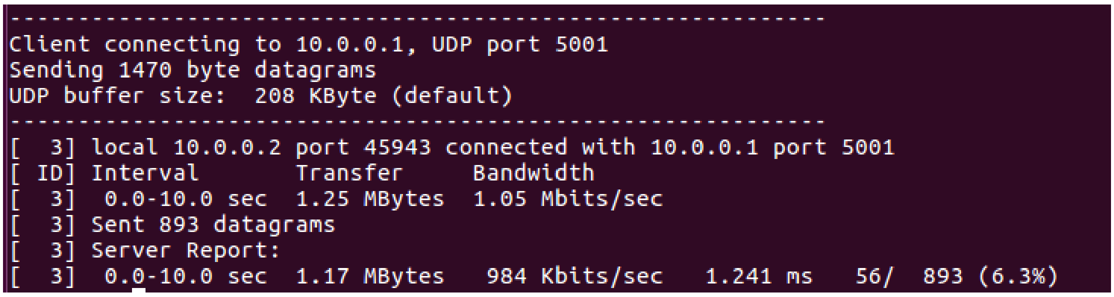
   5.ryu-manager: use ryu-manager run the file

   6.--observe-links: show the information of links

   7.mn: the defaulted topology is minimal

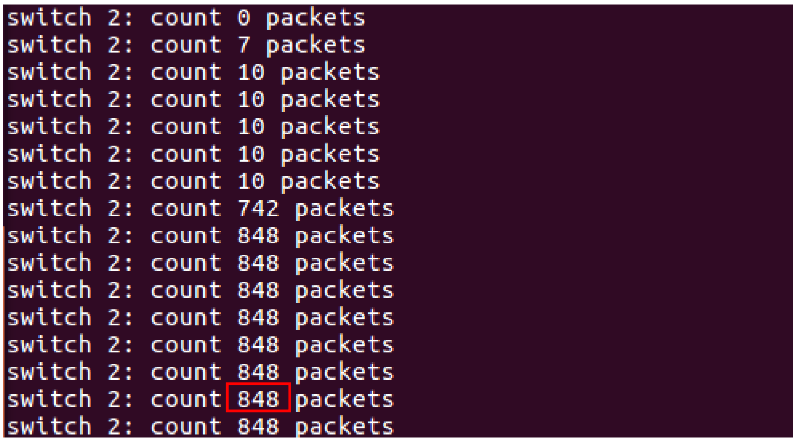
   8.tc:control speed

3.Screeshot:



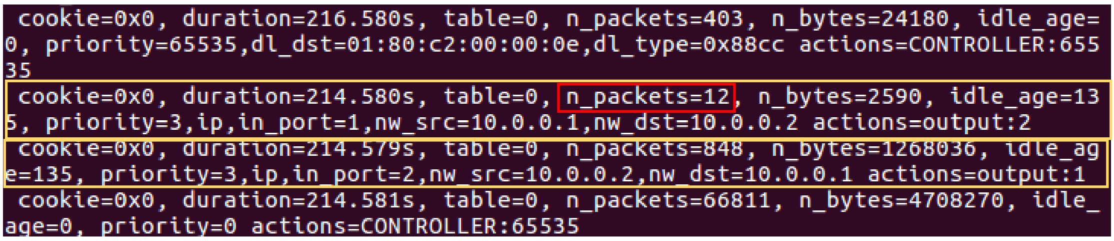
```
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 45943 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.17 MBytes   984 Kbits/sec   1.241 ms   56/  893 (6.3%)
```

(the bandwidth with SimpleController.py)



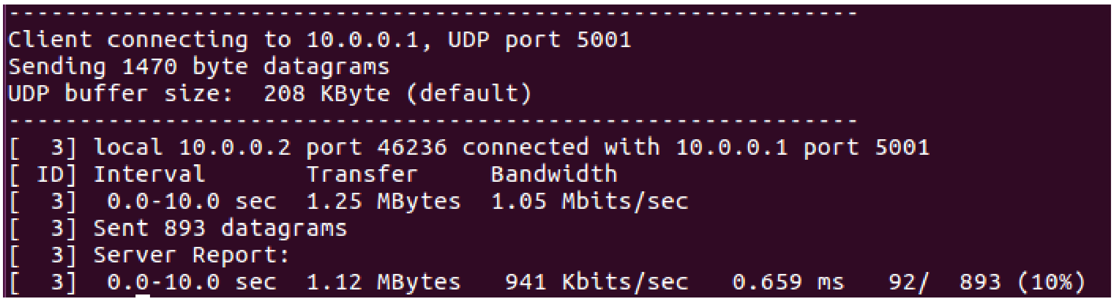
```
switch 2: count 0 packets
switch 2: count 7 packets
switch 2: count 10 packets
switch 2: count 10 packets
switch 2: count 10 packets
switch 2: count 10 packets
switch 2: count 10 packets
switch 2: count 742 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
switch 2: count 848 packets
```

(no more change in the number of packets)



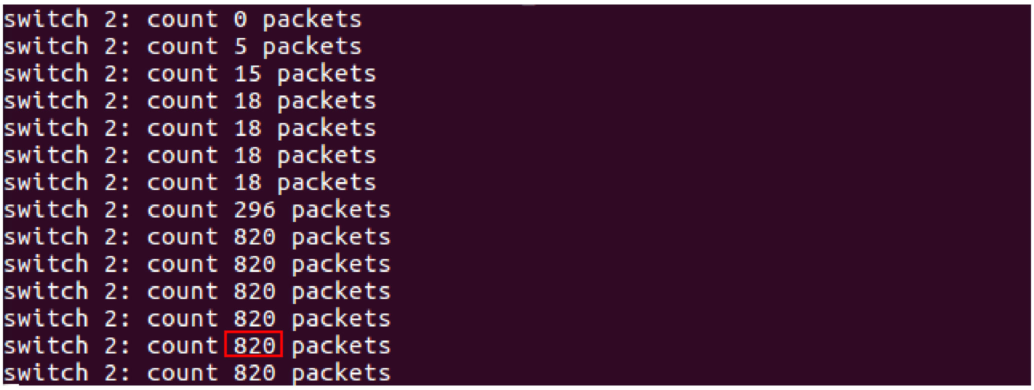
```
 cookie=0x0, duration=216.580s, table=0, n_packets=403, n_bytes=24180, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=214.580s, table=0, n_packets=12, n_bytes=2590, idle_age=13
5, priority=3,ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=214.579s, table=0, n_packets=848, n_bytes=1268036, idle_ag
e=135, priority=3,ip,in_port=2,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=214.581s, table=0, n_packets=66811, n_bytes=4708270, idle_
age=0, priority=0 actions=CONTROLLER:65535
```

(forwarding rules on s2)



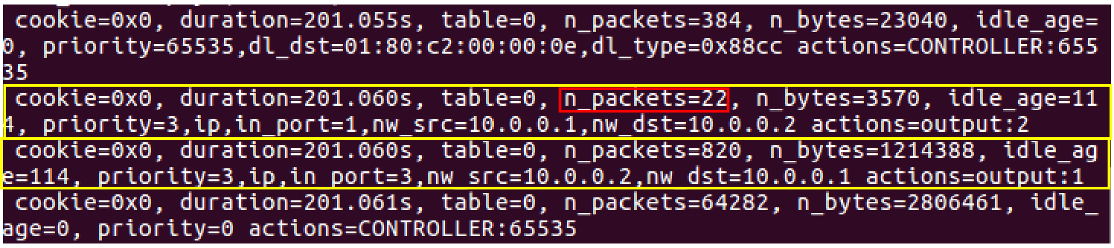
```
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 46236 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.12 MBytes   941 Kbits/sec   0.659 ms   92/  893 (10%)
```

(the bandwidth with controller1.py)



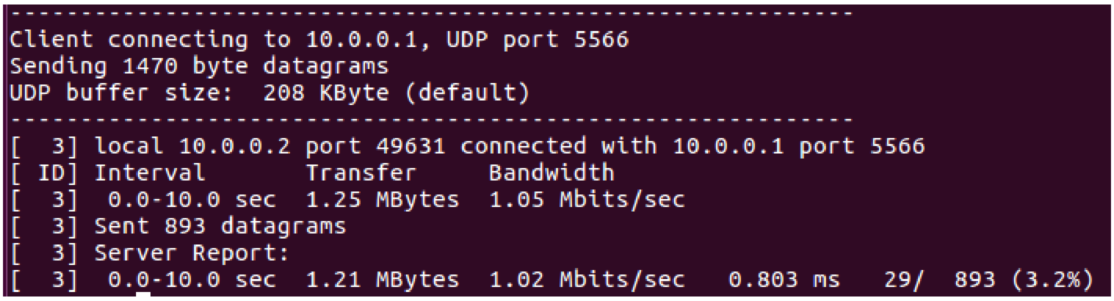
```
switch 2: count 0 packets
switch 2: count 5 packets
switch 2: count 15 packets
switch 2: count 18 packets
switch 2: count 18 packets
switch 2: count 18 packets
switch 2: count 18 packets
switch 2: count 296 packets
switch 2: count 820 packets
switch 2: count 820 packets
switch 2: count 820 packets
switch 2: count 820 packets
switch 2: count 820 packets
switch 2: count 820 packets
```

(no more change in the number of packets)



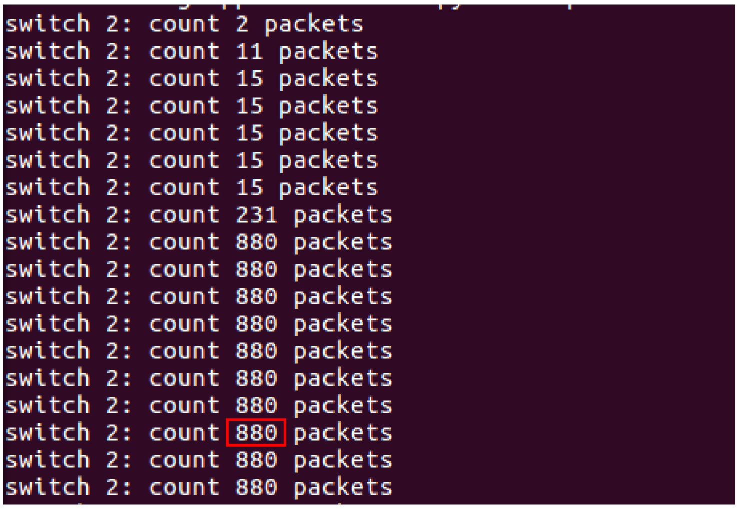
```
 cookie=0x0, duration=201.055s, table=0, n_packets=384, n_bytes=23040, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=201.060s, table=0, n_packets=22, n_bytes=3570, idle_age=11
4, priority=3,ip,in_port=1,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=201.060s, table=0, n_packets=820, n_bytes=1214388, idle_ag
e=114, priority=3,ip,in_port=3,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=201.061s, table=0, n_packets=64282, n_bytes=2806461, idle_
age=0, priority=0 actions=CONTROLLER:65535
```

(forwarding rules on s2)



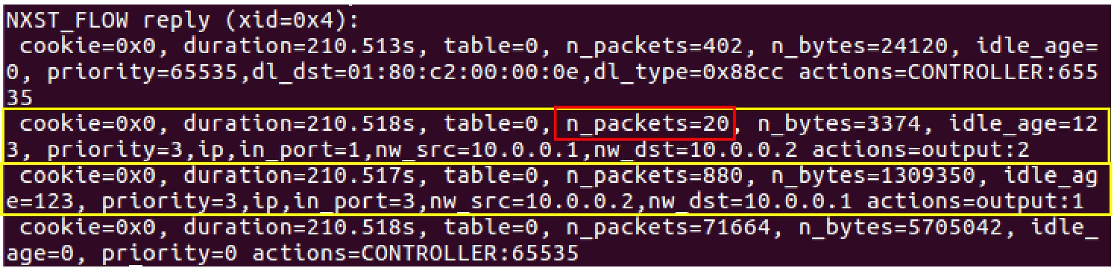
```
------------------------------------------------------------
Client connecting to 10.0.0.1, UDP port 5566
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 49631 connected with 10.0.0.1 port 5566
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.21 MBytes  1.02 Mbits/sec   0.803 ms   29/  893 (3.2%)
```

(the bandwidth with controller2.py)



```
switch 2: count 2 packets
switch 2: count 11 packets
switch 2: count 15 packets
switch 2: count 15 packets
switch 2: count 15 packets
switch 2: count 15 packets
switch 2: count 15 packets
switch 2: count 231 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
switch 2: count 880 packets
```

(no more change in the number of packets)



```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=210.513s, table=0, n_packets=402, n_bytes=24120, idle_age=
0, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x0, duration=210.518s, table=0, n_packets=20, n_bytes=3374, idle_age=12
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=210.517s, table=0, n_packets=880, n_bytes=1309350, idle_ag
e=123, priority=3,ip,in_port=3,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=210.518s, table=0, n_packets=71664, n_bytes=5705042, idle_
age=0, priority=0 actions=CONTROLLER:65535
```

(forwarding rules on s2)

Part2:Handling flow-removed events

1. Explain your code as detail as possible: 首先，將SimpleController.py、controller1.py、controller2.py 的 forwarding rule 都放進 switch_features_handler，依照 priority 1、2、3 區分三者，hard_time 則按照題目要求分別設定，接著利用 flow_removed_handler 中的 msg.packet_count 和 msg.byte_count 計算出剩下的頻寬，當確定三者都有輸出後，再透過比較頻寬大小的判斷式輸出 path 1、path 2、path 3，決定哪個是最佳路徑後，利用 self.addflow，讓最佳路徑能夠繼續傳送。

Part3:Problems encountered

1. Problem you met while doing this lab:
一直不知道為什麼 Cotroller 的頻寬跑出來會對不上 forward rule 應該要有的答案，後來才知道是因為我先 ping 過再跑，所以會有比起原本更多的封包傳送，導致數字都會略大;更改 controller1.py 和 controller2.py 時，也因為沒有注意到需要更改 parser.OFPActionOutput 而一直沒辦法跑出答案，最後的 AdaptiveController 更是因為沒有接觸過寫相關程式的經

驗而花費很多時間研究網路上相關資料，關於在 hard_timeout 之前都沒有封包傳送的情況不知如何下手。

2. **Any advices:**投影片中對於 controller2.py 在執行時是否需要用 5566 的 port 沒有明確說明。

Discussion

1. **Describe the differences between packet-in and packet-out in detail:** packet-in 是 asynchronous，當收到封包時，會由 switch 發出訊息，將封包轉送給 controller。Packet-out 是 controller-to -switch，當接收到來自 controller 的封包時，將其轉送到指定的 port。
2. **What is “table-miss” in SDN:** flow table 找不到相對應的 flow entry。如果發生 table-miss，可能會被直接丟掉、封裝成 packet-in 的訊息，或者繼續轉發。
3. **Why is "(app_manager.RyuApp)" adding after the declaration of class in SimpleController.py:** 用來加載 Ryu 相關的 app，接收發送過來的訊息。
4. **What is the meaning of “datapath” in SimpleController.py:** 為了運用 OpenFlow 的 topo 裡的 switch。
5. **Why need to set "eth_type=0x0800" in the flow entry:** 要使用 IPv4。
6. **Compare the differences between the iPerf results of SimpleController.py,controller1.py and controller2.py.Which forwarding rule is better:**

   SimpleController.py : 0.0-10.0 sec 1.17 MBytes 984Kbits/sec 1.241ms 56/ 893(6.3%) 10 秒內傳了 1.17 MBytes

   平均傳送速度(頻寬) : 984 Kbits/sec

   傳輸延遲 : 1.241ms

   傳送 893 個 datagram，遺失 56 個 datagram => 6.3%遺失率

   Controller1.py :

   0.0-10.0 sec 1.12 MBytes 941 Kbits/sec 0.659ms 92/ 893(10%)

   10 秒內傳了 1.12 Mbytes

   平均傳送速度(頻寬) : 941 Kbits/sec

   傳輸延遲 : 0.659ms

   傳送 893 個 datagram，遺失 92 個 datagram => 10%遺失率

Controller2.py :

0.0-10.0 sec 1.21 MBytes 1.02Mbits/sec 0.803ms 29/ 893(3.2%)

10秒內傳了1.21 Mbytes

平均傳送速度(頻寬) : 1.02 Mbits/sec

傳輸延遲 : 0.803ms

傳送893個datagram，遺失29個datagram => 3.2%遺失率

Controller2.py的遺失率最低，傳輸延遲也只比controller1多0.144ms，因此controller2的路線規劃較好，因此h2從s4到s1到s3，再經過s2到h1較好。