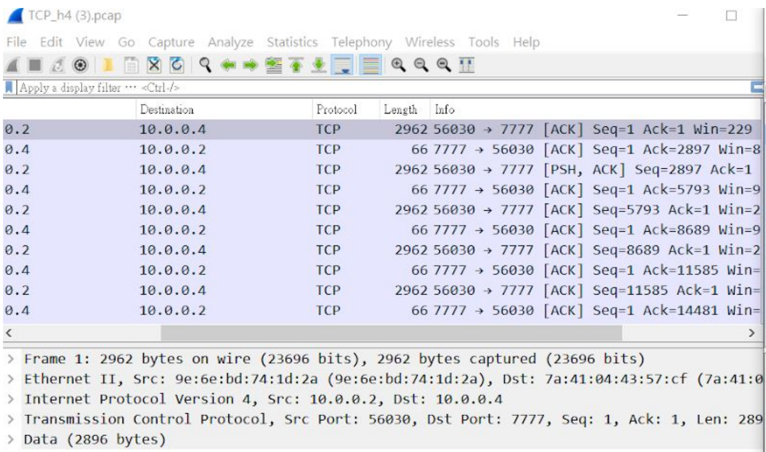
計網概Lab1_0816080 許舒茵

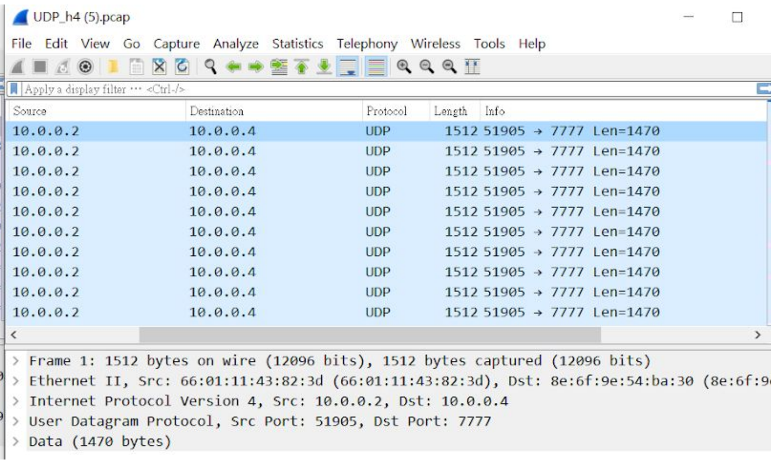
- **Describe each step and how to run your program**

1.安裝MobaXterm及Wireshark
2.打開SSH，輸入ip位址:140.113.195.69 建立環境
3.下載github上的範例檔案，試跑topo.py **(run: sudo python topo.py)**
4.照著圖修改topo_TCP.py及topo_UDP.py，用iperf指令創建各host之間的flow，並運行py檔 **(run: sudo python topo_TCP.py sudo python topo_UDP.py)**
5.到out資料夾下載.pcap後用Wireshark檢查有沒有成功
6.參考parser.py的指令寫computeRate.py，讀檔進computeRate.py後計算 **(run: sudo python computeRate.py ../out/TCP_h3.pcap ../out/TCP_h4.pcap ../out/UDP_h3.pcap ../out/UDP_h4.pcap)**
7.用Wireshark開始.pcap檢查答案

- **Describe your observations from the results in this lab**

TCP傳輸時是雙向傳輸，UDP則是單向





- **What does each iPerf command you used mean?**

```
h3.cmd("tcpdump -w ../out/TCP_h3.pcap &")
# Create flow via iperf
print("create flow via iperf")
# TCP flow
h3.cmd("iperf -s -i 1 -t 5 -p 7777 > ../out/TCP_s_h3_1.txt &")
h1.cmd("iperf -c " + str(h3.IP()) + " -i 1 -t 5 -p 7777 > ../out/TCP_c_h1_1.txt &")
```

-w 指定TCP窗口大小，預設8KB
-s host以server模式啟動
-c host以client模式啟動，host是server端地址
-t 測試時間，預設10秒
-p 指定服務器端使用的端口或是客戶端所連接的端口 (此處為7777)
- ../out/TCP_s_h3_1.txt 把command輸出寫進txt檔中

```
h3.cmd("iperf -s -u -i 1 -t 5 -p 7777 > ../out/UDP_s_h3_1.txt &")
h1.cmd("iperf -c " + str(h3.IP()) + " -u -i 1 -t 5 -p 7777 > ../out/UDP_c_h1_1.txt &")
```

-i sec以秒為單位顯示報告間隔

-u 使用UDP協議

• What is your command to filter each flow in Wireshark?

TCP_h3.pcap:
    tcp.port==7777
    tcp.port==8888
TCP_h4.pcap:
    tcp.port==7777
UDP_h3.pcap:
    udp.port==7777
    udp.port==8888
UDP_h4.pcap:
    udp.port==7777

• Show the results of computeRate.py and statistics of Wireshark

Result of computeRate.py:

```
--- TCP ---
('Flow1(h1->h3):    ', 0.69941092860446, '    Mdps')
('Flow2(h1->h3):    ', 0.5814209236220418, '    Mdps')
('Flow3(h2->h4):    ', 0.733584373265731, '   Mdps')
--- UDP ---
('Flow1(h1->h3):    ', 0.6561931416854624, '    Mdps')
('Flow2(h1->h3):    ', 0.6541105114184229, '    Mdps')
('Flow3(h2->h4):    ', 0.7068410327833352, '   Mdps')
```

statistics of Wireshark:

TCP_h3.pcap

tcp.port==7777

Measurement	Captured	Displayed	Marked
Packets	230	118 (51.3%)	—
Time span, s	2.025	1.994	—
Average pps	113.6	59.2	—
Average packet size, B	1408	1502	—
Bytes	323856	177216 (54.7%)	0
Average bytes/s	159 k	88 k	—
Average bits/s	1279 k	710 k	—

TCP_h3.pcap

tcp.port==8888

Measurement	Captured	Displayed	Marked
Packets	230	98 (42.6%)	—
Time span, s	2.025	1.958	—
Average pps	113.6	50.0	—
Average packet size, B	1408	1484	—
Bytes	323856	145476 (44.9%)	0
Average bytes/s	159 k	74 k	—
Average bits/s	1279 k	594 k	—

TCP_h4.pcap

tcp.port==7777

Measurement	Captured	Displayed	Marked
Packets	139	125 (89.9%)	—
Time span, s	1.988	1.988	—
Average pps	69.9	62.9	—
Average packet size, B	1349	1491	—
Bytes	187518	186354 (99.4%)	0
Average bytes/s	94 k	93 k	—
Average bits/s	754 k	749 k	—

UDP_h3.pcap

udp.port==7777

Measurement	Captured	Displayed	Marked
Packets	230	109 (47.4%)	—
Time span, s	2.009	2.009	—
Average pps	114.5	54.2	—
Average packet size, B	1431	1512	—
Bytes	329186	164808 (50.1%)	0
Average bytes/s	163 k	82 k	—
Average bits/s	1310 k	656 k	—

UDP_h3.pcap
udp.port==8888

Measurement	Captured	Displayed	Marked
Packets	230	108 (47.0%)	—
Time span, s	2.009	1.997	—
Average pps	114.5	54.1	—
Average packet size, B	1431	1512	—
Bytes	329186	163296 (49.6%)	0
Average bytes/s	163 k	81 k	—
Average bits/s	1310 k	654 k	—

UDP_h4.pcap
udp.port==7777

Measurement	Captured	Displayed	Marked
Packets	129	116 (89.9%)	—
Time span, s	1.985	1.985	—
Average pps	65.0	58.4	—
Average packet size, B	1368	1512	—
Bytes	176474	175392 (99.4%)	0
Average bytes/s	88 k	88 k	—
Average bits/s	711 k	706 k	—

• What you have learned from this lab?

1.更熟悉Github pull push的用法
2.學到新工具MobaXterm及Wireshark，自己建立模擬的環境
3.學習iperf指令的寫法，建立模擬的host跟flow並用Wireshark debug
4.學會Linux指令和在Mininet運作python，還有parser裡提供的指令
5.更清楚TCP跟UDP的差別
6.了解關於封包傳送時間跟大小的關係(ex:取time跟size要如何取)
7.了解封包走不同路徑的狀況
8.知道Mdps怎麼換算
9.用Wireshark分析.pcap，了解封包傳送的狀況，並利用filter把不同flow的封包分開

• What difficulty you have met in this lab?

1.剛接觸新工具有點難上手，先上網查了各個操作
2.iperf指令不熟悉，剛開始寫topo.py時卡很久
3.不知道h1 h3的兩條flow必須寫不一樣的port，導致後面出錯

4.寫topo_TCP跟topo_UDP的時候多寫了一條 `h4.cmd("tcpdump -w ../out/TCP_h4.pcap &")`導致TCP_h4.pcap被新的空檔案覆蓋，後面跑computeRate.py的時候flow也會是空的
5.剛開始想跑computeRate.py的時候不知道後面要寫檔案位址讓他讀檔，一直出現sys.arg[1]out of range的錯誤，本來以為是溢位後來才查到是沒有讀檔案
6.算rate時數據一直不太對，修改了一下算式的寫法誤差才變小
7.在parser裡學了會用到的指令，唯獨不知道要怎麼抓time，問很久才找到
8.剛用Wireshark檢查數據時沒有把port分清楚，以為是computeRate出錯，其實就算只有一個flow也要用filter過濾一下