Lab1 Report

Sniffing and Measurements

Student: 陳孟楷

Student ID: 113550021

November 2024

1. Part 1

1.1 Output Result from compute.py

```
Reading pcap file: udp1M.pcap ...
Calculating total bytes for 21.09 seconds...
Total bytes result:
Flow1 (UDP on port 7778): 2479680 Bytes
Flow2 (TCP on port 7777): 126980 Bytes
Flow3 (TCP on port 7776): 0 Bytes

bps result:
Flow1 (UDP on port 7778): 940909.65 bps
Flow2 (TCP on port 7777): 48835.04 bps
Flow3 (TCP on port 7776): -0.00 bps

Reading pcap file: udp0.5M.pcap ...
Calculating total bytes for 27.20 seconds...
Total bytes result:
Flow1 (UDP on port 7778): 1363824 Bytes
Flow2 (TCP on port 7777): 1424512 Bytes
Flow3 (TCP on port 7776): 0 Bytes

bps result:
Flow1 (UDP on port 7778): 529814.07 bps
Flow2 (TCP on port 7777): 511049.31 bps
Flow3 (TCP on port 7776): -0.00 bps

Reading pcap file: udp0.5M_2.pcap ...
Calculating total bytes for 22.94 seconds...
Total bytes result:
Flow1 (UDP on port 7778): 1363824 Bytes
Flow2 (TCP on port 7777): 677362 Bytes
Flow3 (TCP on port 7776): 825866 Bytes

bps result:
Flow1 (UDP on port 7778): 523480.30 bps
Flow2 (TCP on port 7777): 245185.75 bps
Flow3 (TCP on port 7776): 288059.12 bps

All graphs have been saved.
```

Figure 1. Terminal of compute.py

1.2 Bar Graphs Generated by compute.py

1.2.1 Scenario 1

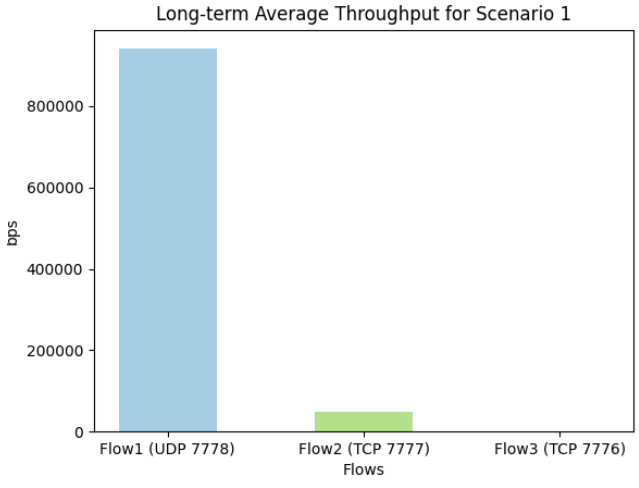

Figure 2. Bar Graph of Scenario 1

1.2.2 Scenario 2

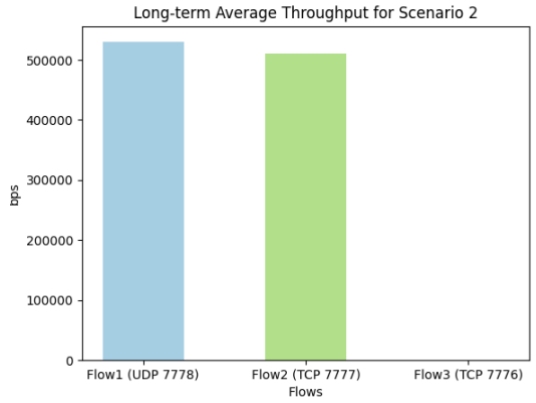

Figure 3. Bar Graph of Scenario 2

1.2.3 Scenario 3

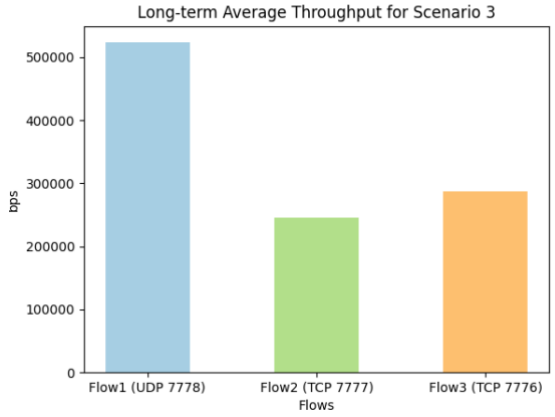

Figure 4. Bar Graph of Scenario 3

1.3 Wireshark Statistics Results

1.3.1 Scenario 1

1.3.1.1 UDP on Port 7778

Measurement	Captured	Displayed	Marked
Packets	1799	1640 (91.2%)	—
Time span, s	21.091	21.083	—
Average pps	85.3	77.8	—
Average packet size, B	1452	1512	—
Bytes	2612322	2479680 (94.9%)	0
Average bytes/s	123 k	117 k	—
Average bits/s	990 k	940 k	—

Figure 5. UDP on 7778 of Scenario 1

1.3.1.2 TCP on port 7777

Measurement	Captured	Displayed	Marked
Packets	1799	80 (4.4%)	—
Time span, s	21.091	20.801	—
Average pps	85.3	3.8	—
Average packet size, B	1452	1587	—
Bytes	2612322	126980 (4.9%)	0
Average bytes/s	123 k	6104	—
Average bits/s	990 k	48 k	—

Figure 6. TCP on 7777 of Scenario 1

1.3.2 Scenario 2

1.3.2.1 UDP on Port 7778

Measurement	Captured	Displayed	Marked
Packets	2763	902 (32.6%)	—
Time span, s	27.196	20.593	—
Average pps	101.6	43.8	—
Average packet size, B	1036	1512	—
Bytes	2861098	1363824 (47.7%)	0
Average bytes/s	105 k	66 k	—
Average bits/s	841 k	529 k	—

Figure 7. UDP on 7778 of Scenario 2

1.3.2.2 TCP on Port 7777

Measurement	Captured	Displayed	Marked
Packets	2763	942 (34.1%)	—
Time span, s	27.196	22.299	—
Average pps	101.6	42.2	—
Average packet size, B	1036	1512	—
Bytes	2861098	1424512 (49.8%)	0
Average bytes/s	105 k	63 k	—
Average bits/s	841 k	511 k	—

Figure 8. TCP on 7777 of Scenario 2

1.3.3 Scenario 3

1.3.3.1 UDP on Port 7778

Measurement	Captured	Displayed	Marked
Packets	2722	902 (33.1%)	—
Time span, s	22.936	20.842	—
Average pps	118.7	43.3	—
Average packet size, B	1076	1512	—
Bytes	2930032	1363824 (46.5%)	0
Average bytes/s	127 k	65 k	—
Average bits/s	1021 k	523 k	—

Figure 9. UDP on 7778 of Scenario 3

1.3.3.2 TCP on Port 7777

Measurement	Captured	Displayed	Marked
Packets	2722	433 (15.9%)	—
Time span, s	22.936	22.101	—
Average pps	118.7	19.6	—
Average packet size, B	1076	1564	—
Bytes	2930032	677362 (23.1%)	0
Average bytes/s	127 k	30 k	—
Average bits/s	1021 k	245 k	—

Figure 10. TCP on 7777 of Scenario 3

1.3.3.3 TCP on Port 7776

Measurement	Captured	Displayed	Marked
Packets	2722	533 (19.6%)	—
Time span, s	22.936	22.936	—
Average pps	118.7	23.2	—
Average packet size, B	1076	1549	—
Bytes	2930032	825866 (28.2%)	0
Average bytes/s	127 k	36 k	—
Average bits/s	1021 k	288 k	—

Figure 11. TCP on 7776 of Scenario 3

2. Part 2

- **What are the iPerf commands you used in Task 2, scenario 2&3? What do they mean?**

  At first, I followed the steps of Scenario 1 to start the server listening UDP on 7778 and TCP on 7777 in scenario 2. In scenario 3, I followed the similar steps to start the server listening UDP on 7778, TCP on 7777 and another TCP on 7776.

  For UDP, I inputted "**iperf -s -u -i 1 -p 7778**" (**-s** means running iPerf in server mode, **-u** means using UDP, **-i 1** means setting the interval time as 1 second, **-p 7778** means using port 7778 for the server to listen and the client to connect).

  For TCP, I inputted "i**perf -s -i 1 -p 7777**" in scenario 2 and add "**iperf -s -i 1 -p 7776**" in scenario 3 (**-s -i 1** are the same as the UDP, **-p 7777** and **-p 7776** means using port 7777/7776 for the server to listen and the client to connect).

  Then I used iPerf to generate flows.

  For Senerio2 , I inputted "**iPerf -c server -u -i 1 -t 20 -b 0.5M -p 7778 & iPerf -c server -i 1 -t 20 -p 7777**" (**-c** means running iPerf in client mode, **-t 20** means setting 20 seconds to transmit, **-b 0.5M** means setting bandwidth as 0.5Mbits/sec, other commands are the same as mentioned above).

  For Scenario 3, I inputted "**iPerf -c server -u -i 1 -t 20 -b 0.5M -p 7778 & iPerf -c server -i 1 -t 20 -p 7777 & iPerf -c server -i 1 -t 20 -p 7776**" (commands are all the same as mentioned above).

  The key difference between scenario 2 and 3 is the third TCP connection. For this TCP, we need to set one more port (7776) to listen and connect. Also, we need to use iPerf to generate one more TCP flow on 7776.

- **What is your command to filter each flow in Wireshark**

  I used **the protocol (udp / tcp)**, **destination port (dstport)**, and the **port number (7778 / 7777 / 7776)** to filter every flow in Wireshark. In scenario1 and 2, I used "**udp.dstport == 7778**" to find all UDP transfers and "**tcp.dstport == 7777**" to find all TCP transfers. In scenario 3, I used "**udp.dstport == 7778**" to find all UDP transfers, "**tcp.dstport == 7777**" to find the transfers on first TCP connection, and "**tcp.dstport == 7776**" to find the transfers on the second TCP connection.

- **In these three scenarios, we set the client's rate limit to 1Mbps, and when generating flows with iPerf, we assigned different transmission rates to UDP. Why do UDP and TCP have different throughput results at the end? Please explain.**

  Because UDP does not need to do acknowledgement back and forth between the client and the server, but TCP does, TCP needs more time to send a fixed numbers of packets than UDP (i.e. The throughput of UDP is larger than TCP).

  By comparing scenario 1 and 2, it shows that if we set the transmitting rate of UDP too high, then the throughput of TCP would become much smaller than UDP, because UDP would occupy almost the whole available bandwidth.

  By comparing scenario 2 and 3, it shows that if we set 2 TCP flows at the same time (scenario 3), the throughput of each TCP flow would become smaller than the situation of only using 1 TCP flow (scenario 2). However, the throughput of UDP may be influenced only a little bit.

3. Bonus

- **What have you learned from this lab?**

  In this lab, I learned how to simulate the client and the server by using Docker. Also, I learned how to use iPerf to set the server listen on designated ports and generate both UDP and TCP flows. In Wireshark, I learned how to filter specific packets using different commands and analyze the statistics generated by Wireshark. I really learned a lot from this lab.

- **What difficulty have you met in this lab?**

  At first, I found my bar graphs are a little bit different from others' which are posted on Teams Lab channel. I discussed this situation with my roommates, and we finally found the mistake was that I transmitted the UDP and TCP separately. In fact, they were supposed to be transmitted at the same time, which could test whether UDP owned higher throughput or TCP. Besides, while I was filtering packets in Wireshark, the number of bytes it showed was a little bit different from the results of compute.py. I googled it and found that I should use **udp.dstport/tcp.dstport** instead of **udp.port/tcp.port** since the latter refers to both source port and destination port which may include some packets that we don't want.