Lab1 Report

Sniffing and Measurements

Student: 陳孟楷

Student ID: 113550021

November 2024

1. Part 1
   1. Output Result from compute.py

一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

自動產生的描述

Figure 1. Terminal of compute.py

* 1. Bar Graphs Generated by compute.py
     1. Scenario 1

一張含有 文字, 螢幕擷取畫面, 圖表, 陳列 的圖片

自動產生的描述

Figure 2. Bar Graph of Scenario 1

* + 1. Scenario 2

一張含有 文字, 螢幕擷取畫面, 圖表, Rectangle 的圖片

自動產生的描述

Figure 3. Bar Graph of Scenario 2

* + 1. Scenario 3

一張含有 文字, 螢幕擷取畫面, 圖表, Rectangle 的圖片

自動產生的描述

Figure 4. Bar Graph of Scenario 3

* 1. Wireshark Statistics Results
     1. Scenario 1
        1. UDP on Port 7778

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 5. UDP on 7778 of Scenario 1

* + - 1. TCP on port 7777

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 6. TCP on 7777 of Scenario 1

* + 1. Scenario 2
       1. UDP on Port 7778

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 7. UDP on 7778 of Scenario 2

* + - 1. TCP on Port 7777

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 8. TCP on 7777 of Scenario 2

* + 1. Scenario 3
       1. UDP on Port 7778

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 9. UDP on 7778 of Scenario 3

* + - 1. TCP on Port 7777

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 10. TCP on 7777 of Scenario 3

* + - 1. TCP on Port 7776

一張含有 文字, 收據, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Figure 11. TCP on 7776 of Scenario 3

1. 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.

1. **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.