1. How I run my program

Task 2:

After setting up the environment in task 1, I first create a new text editor in terminal using vim and add some new hosts and links according to the problem by modifying the original code inside the topo example.

Task 3:

I did almost the same as the example code when generating TCP flow, but I modified some iPerf command options as UDP requires to assign its UDP property and bandwidth, so I add two command: -u and -b 5M(as the task says that the bandwidth is assigned to 5Mbps)

Task 4:

As it requires to read the flow in the packet, I opened the pcap file in the wireshark and found that I only had to extract the packets from those delivered to port 7777 and calculate their time, so I wrote

some function to filter out the packet that is not delivered to port 7777 or either not send through TCP/UDP protocol. And I calculate the size of all the packets using a for loop and thus calculate the throughput of the flows.

2. My observations from the results

In pcap files, I saw there's all packets are delivered with flags(some have one and some have multiple), and even we filter the protocol by either TCP or UDP and a particular port 7777, there might also include some packets with MDNS or ICMP protocol with port that is unreachable or other than 7777. Moreover, the calculated throughput is a little bit different from the statistics wireshark measured.

3. Questions below:

- The meaning of the commands I used in iPerf:
  - -s : server
  - -c : client
  - -i : set the transmission time interval as a unit

- -t : set the total transmission time
- -p : set the communication port
- -u : use UDP protocol
- -b : set particular bandwidth(as UDP use 1Mbps as default)

- My command to filter each flow in Wireshark:
  - To filter TCP_h3, I use “tcp and ip.src==10.0.0.2 and ip.dst==10.0.0.3”
  - To filter TCP_h4, I use “tcp and ip.src==10.0.0.1 and ip.dst==10.0.0.4”
  - To filter UDP_h3, I use “udp and ip.src==10.0.0.2 and ip.dst==10.0.0.3”
  - To filter UDP_h4, I use “udp and ip.src==10.0.0.1 and ip.dst==10.0.0.4”
- The results of computeRate.py and statistics of Wireshark

computeRate.py:

```
--- TCP ---
Flow1(h1->h4): 1.8923334688701585 Mbps
Flow2(h2->h3): 2.8717175341071317 Mbps
--- UDP ---
Flow1(h1->h4): 1.81644283518394 Mbps
Flow2(h2->h3): 3.001020442225131 Mbps
```

Wireshark TCP Flow1:

Statistics

Measurement	Captured	Displayed	Marked
Packets	910	446 (49.0%)	—
Time span, s	16.635	5.057	—
Average pps	54.7	88.2	—
Average packet size, B	1387	2754	—
Bytes	1261834	1228388 (97.3%)	0
Average bytes/s	75k	242k	—
Average bits/s	606k	1943k	—

Wireshark TCP Flow2:

Statistics

Measurement	Captured	Displayed	Marked
Packets	1332	671 (50.4%)	—
Time span, s	16.539	5.041	—
Average pps	80.5	133.1	—
Average packet size, B	1430	2770	—
Bytes	1905066	1858638 (97.6%)	0
Average bytes/s	115k	368k	—
Average bits/s	921k	2949k	—

Wireshark UDP Flow1:

Statistics

Measurement	Captured	Displayed	Marked
Packets	939	857 (91.3%)	—
Time span, s	16.476	5.444	—
Average pps	57.0	157.4	—
Average packet size, B	1389	1496	—
Bytes	1304676	1281954 (98.3%)	0
Average bytes/s	79k	235k	—
Average bits/s	633k	1883k	—

Wireshark UDP Flow2:

Statistics

Measurement	Captured	Displayed	Marked
Packets	1468	1386 (94.4%)	—
Time span, s	16.476	5.326	—
Average pps	89.1	260.2	—
Average packet size, B	1433	1501	—
Bytes	2103602	2080880 (98.9%)	0
Average bytes/s	127k	390k	—
Average bits/s	1021k	3125k	—

- The throughput did match the bottleneck throughput. As in flow 1, the bottleneck throughput is 2Mbps and in flow 2, the bottleneck throughput is 5Mbps in the link between two switches. However, this 5Mbps is shared by both flows, so flow1+flow2 should not exceed 5Mbps, and the result did match the bottleneck throughput.
- I did observe the same throughput form TCP and UDP but neither flows can equally share the bandwidth.

4. Bonus

- As I’ve never written Python code before, I first tried to learn something in the example codes(topo.py , parser.py). It is ok for me to just altered the code of the example to complete task 2 and task 3. However, when I headed to task 4, I knew that I at least had to learn how to write a for loop in python so I surfed the Internet to learn

this. Moreover, I saw that there are flags in each packet in the pcap file, so I went search for the meaning of those flags(fin for finish and ack for acknowledge etc.). Most importantly, after constructing a miniature packet flows topology, I now have deeper acknowledgement of packet flows rather than a mere abstract concept.

- The difficulty I met is that I didn't know why the estimated timestamp of my topo_UDP is up to 15 sec. After doing research and discussing with my classmates, I still couldn't figure it out, so I asked TA for the possible reason. It turned out that I didn't filter the last packet with the port condition, so I over-measured the packets with MDNS protocol. After adding the port condition, the calculated avg throughput is correct.