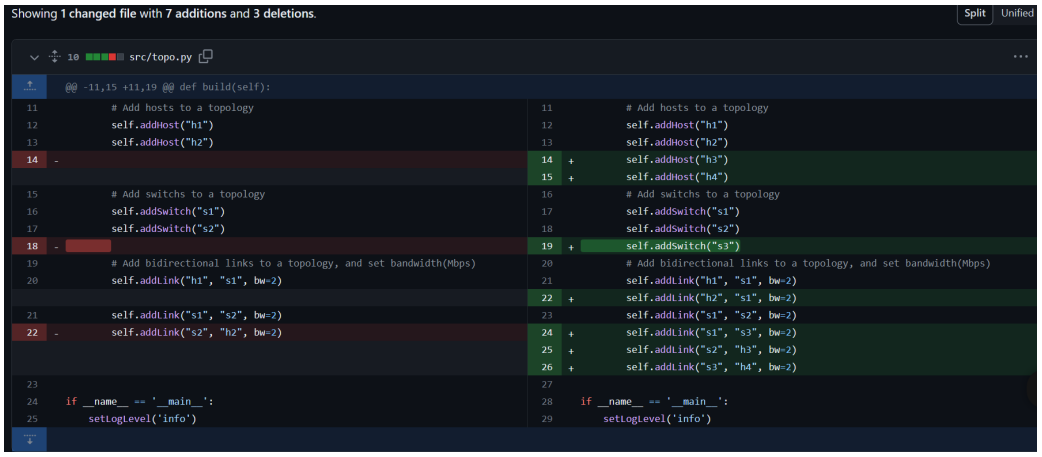
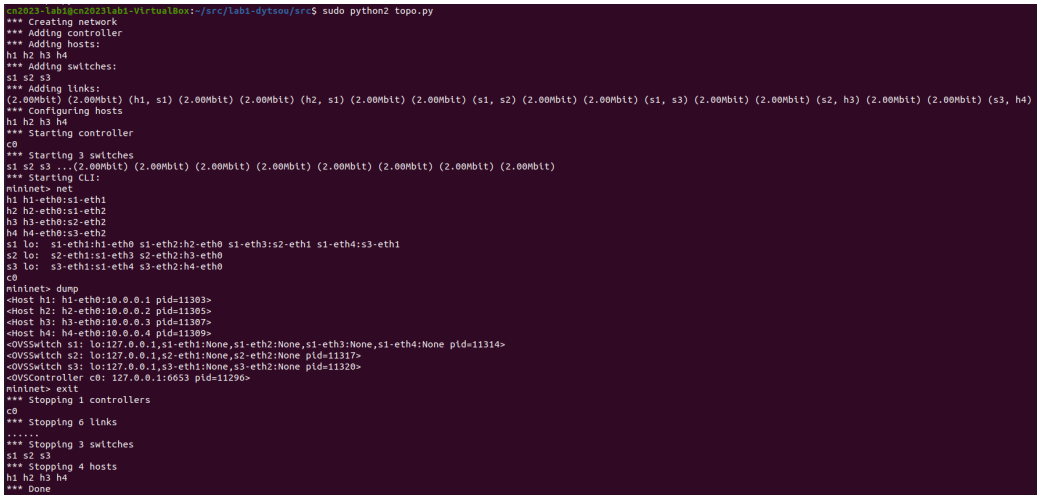
CN-Lab1 Report

- Describe each step and how to run your program
  - Task2
    1. un `nano topo.py` to modify topo.py .



    2. Run `sudo python2 topo.py` to see the result.



    3. run `sudo mn -c` to clean uo the environment
  - Task 3
    1. Make 2 copies topo_TCP.py, topo_UDP.py from topo.py
    2. Modify topo_TCP.py, topo_UDP.py

3. Run `sudo python2 topo_TCP.py` and sudo `python2 topo_UDP.py` to generate TCP flows and UDP flows, and generate the following files
    - ..out/TCP_c_h1_1.txt
    - ..out/TCP_c_h1_2.txt
    - ../out/TCP_c_h2.txt
    - ../out/TCP_h3.pcap
    - ..out/TCP_h4.pcap
    - ../out/TCP_s_h3_1.txt
    - ../out/TCP_s_h3_2.txt
    - ../out/TCP_s_h4.txt
    - ../out/UDP_c_h1_1.txt
    - ../out/UDP_c_h1_2.txt
    - ../out/UDP_c_h2.txt
    - ../out/UDP_h3.pcap
    - ../out/UDP_h4.pcap
    - ../out/UDP_s_h3_1.txt
    - ../out/UDP_s_h3_2.txt
    - ../out/UDP_s_h4.txt

- Task 4
    1. Run `sudo python3 parser.py ../out/TCP_h3.pcap` and observe the result to know how to modify the code.
    2. Write in computeRate.py.
    3. Run `sudo python3 computeRate.py` and get the following results

```
cn2023-lab1@cn2023lab1-VirtualBox:~/src/lab1-dytsou/src$ sudo python3 computeRate.py
--- TCP ---
Flow1(h1->h3): 1.0079356307085254 Mbps
Flow2(h1->h3): 0.9986187678840517 Mbps
Flow3(h2->h4): 1.9933657148686337 Mbps
--- UDP ---
Flow1(h1->h3): 1.018179186878172 Mbps
Flow2(h1->h3): 1.0181842615473817 Mbps
Flow3(h2->h4): 1.0769857795667783 Mbps
```

- Task 5

1. Open the pcap files using Wireshark
2. Filter each pcap file to get the following results
    - TCP Flow1: Using `tcp.port == 7777` to filter out/TCP_h3.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	915	423 (46.2%)	—
Time span, s	32.110	5.048	—
Average pps	28.5	83.8	—
Average packet size, B	1396	1504	—
Bytes	1276952	636054 (49.8%)	0
Average bytes/s	39 k	125 k	—
Average bits/s	318 k	1007 k	—

TCP Flow1

    - TCP Flow2: Using `tcp.port == 8888` to filter out/TCP_h3.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	915	424 (46.3%)	—
Time span, s	32.110	5.061	—
Average pps	28.5	83.8	—
Average packet size, B	1396	1490	—
Bytes	1276952	631804 (49.5%)	0
Average bytes/s	39 k	124 k	—
Average bits/s	318 k	998 k	—

TCP Flow2

    - TCP Flow3: Using `tcp.port == 7777` to filter out/TCP_h4.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	904	835 (92.4%)	—
Time span, s	5.039	5.039	—
Average pps	179.4	165.7	—
Average packet size, B	1399	1504	—
Bytes	1264895	1255490 (99.3%)	0
Average bytes/s	251 k	249 k	—
Average bits/s	2008 k	1993 k	—

TCP Flow3

- UDP Flow1: Using `udp.port == 7777 and !icmp` to filter out/UDP_h3.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	1004	456 (45.4%)	—
Time span, s	15.907	5.417	—
Average pps	63.1	84.2	—
Average packet size, B	1391	1512	—
Bytes	1396817	689472 (49.4%)	0
Average bytes/s	87 k	127 k	—
Average bits/s	702 k	1018 k	—

UDP Flow1

- UDP Flow2: Using `udp.port == 8888 and !icmp` to filter out/TCP_h3.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	1004	456 (45.4%)	—
Time span, s	15.907	5.417	—
Average pps	63.1	84.2	—
Average packet size, B	1391	1512	—
Bytes	1396817	689472 (49.4%)	0
Average bytes/s	87 k	127 k	—
Average bits/s	702 k	1018 k	—

UDP Flow2

- UDP Flow3: Using `udp.port == 7777 and !icmp` to filter out/TCP_h4.pcap

Statistics

Measurement	Captured	Displayed	Marked
Packets	529	445 (84.1%)	—
Time span, s	25.387	4.998	—
Average pps	20.8	89.0	—
Average packet size, B	1293	1512	—
Bytes	683965	672840 (98.4%)	0
Average bytes/s	26 k	134 k	—
Average bits/s	215 k	1076 k	—

UDP Flow3

- Describe your observations from the results in this lab
  1. Flow1 and Flow2 have exactly same result in TCP, and so do in UDP.
  2. In the case of TCP, Flow3's rate is almost twice as the others; however, Flow3 of UDP only slightly higher than the others.
  3. In my thought, UDP should have higher rate than TCP due to TCP's three-way-shaking causing longer time. Surprisingly, the result is not significant. I guess it is because I didn't do the tests at the same time that makes the result.
  4. All the flows fit the bandwidth, but Flow3 of TCP has a rate really close to the bandwidth.
- Answer the following question in short:
  - What does each iPerf command you used mean?

```
# net.get() is used to get the reference to Mininet host object according
# to the host name with "h1", "h2", "h3", "h4"
h1 = net.get("h1")
h2 = net.get("h2")
h3 = net.get("h3")
h4 = net.get("h4")
```

```
# Use tcpdump to record packet in background
# '-w' for write, '&' for run in background

## topo_TCP.py ##
h3.cmd("tcpdump -w ../out/TCP_h3.pcap &")
```

```
h4.cmd("tcpdump -w ../out/TCP_h4.pcap &")

## topo_UDP.py ##
h3.cmd("tcpdump -w ../out/UDP_h3.pcap &")
h4.cmd("tcpdump -w ../out/UDP_h4.pcap &")
```

```
# '-s' means run iperf as server
# '-c' means run iperf as client
# '-i 1' means iperf report the performance statistics every 1 second
# '-t 5' means iperf test run for 5 seconds
# '-u' means use UDP
# '-p 7777/8888' means the port number used by iperf server or client

## topo_TCP.py ##
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 &")
h3.cmd("iperf -s -i 1 -t 5 -p 8888 > ../out/TCP_s_h3_2.txt &")
h1.cmd("iperf -c " + str(h3.IP()) +
       " -i 1 -t 5 -p 8888 > ../out/TCP_c_h1_2.txt &")
h4.cmd("iperf -s -i 1 -t 5 -p 7777 > ../out/TCP_s_h4.txt &")
h2.cmd("iperf -c " + str(h4.IP()) +
       " -i 1 -t 5 -p 7777 > ../out/TCP_c_h2.txt &")

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

h3.cmd("iperf -s -i 1 -t 5 -u  -p 8888 > ../out/UDP_s_h3_2.txt &")
h1.cmd("iperf -c " + str(h3.IP()) +
       " -i 1 -t 5 -u -p 8888 > ../out/UDP_c_h1_2.txt &")

h4.cmd("iperf -s -i 1 -t 5 -u -p 7777 > ../out/UDP_s_h4.txt &")
h2.cmd("iperf -c " + str(h4.IP()) +
       " -i 1 -t 5 -u -p 7777 > ../out/UDP_c_h2.txt &")
```

- What is your command to filter each flow in Wireshark?
  - `tcp.port == 7777` : only to show TCP packet with port 7777
  - `tcp.port == 8888` : only to show TCP packet with port 8888
  - `udp.port == 7777` : only to show UDP packet with port 7777
  - `udp.port == 8888` : only to show UDP packet with port 8888
  - `!icmp` : not to show packet with prototype == icmp
- Show the results of computeRate.py and statistics of Wireshark
  - Results of computeRate.py

```
cn2023-lab1@cn2023lab1-VirtualBox:~/src/lab1-dytsou/src$ sudo python3 computeRate.py
--- TCP ---
Flow1(h1->h3): 1.0079356307085254 Mbps
Flow2(h1->h3): 0.9986187678840517 Mbps
Flow3(h2->h4): 1.9933657148686337 Mbps
--- UDP ---
Flow1(h1->h3): 1.018179186878172 Mbps
Flow2(h1->h3): 1.0181842615473817 Mbps
Flow3(h2->h4): 1.0769857795667783 Mbps
```

- Statistics of TCP Flow1

Statistics

Measurement	Captured	Displayed	Marked
Packets	915	423 (46.2%)	—
Time span, s	32.110	5.048	—
Average pps	28.5	83.8	—
Average packet size, B	1396	1504	—
Bytes	1276952	636054 (49.8%)	0
Average bytes/s	39 k	125 k	—
Average bits/s	318 k	1007 k	—

TCP Flow1

- Statistics of TCP Flow2

Statistics

Measurement	Captured	Displayed	Marked
Packets	915	424 (46.3%)	—
Time span, s	32.110	5.061	—
Average pps	28.5	83.8	—
Average packet size, B	1396	1490	—
Bytes	1276952	631804 (49.5%)	0
Average bytes/s	39 k	124 k	—
Average bits/s	318 k	998 k	—

TCP Flow2

- Statistics of TCP Flow3

Statistics

Measurement	Captured	Displayed	Marked
Packets	904	835 (92.4%)	—
Time span, s	5.039	5.039	—
Average pps	179.4	165.7	—
Average packet size, B	1399	1504	—
Bytes	1264895	1255490 (99.3%)	0
Average bytes/s	251 k	249 k	—
Average bits/s	2008 k	1993 k	—

TCP Flow3

- Statistic of UDP Flow1

Statistics

Measurement	Captured	Displayed	Marked
Packets	1004	456 (45.4%)	—
Time span, s	15.907	5.417	—
Average pps	63.1	84.2	—
Average packet size, B	1391	1512	—
Bytes	1396817	689472 (49.4%)	0
Average bytes/s	87 k	127 k	—
Average bits/s	702 k	1018 k	—

UDP Flow1

- Statistic of UDP Flow2

Statistics

Measurement	Captured	Displayed	Marked
Packets	1004	456 (45.4%)	—
Time span, s	15.907	5.417	—
Average pps	63.1	84.2	—
Average packet size, B	1391	1512	—
Bytes	1396817	689472 (49.4%)	0
Average bytes/s	87 k	127 k	—
Average bits/s	702 k	1018 k	—

UDP Flow2

- Statistic of UDP Flow3

Statistics

Measurement	Captured	Displayed	Marked
Packets	529	445 (84.1%)	—
Time span, s	25.387	4.998	—
Average pps	20.8	89.0	—
Average packet size, B	1293	1512	—
Bytes	683965	672840 (98.4%)	0
Average bytes/s	26 k	134 k	—
Average bits/s	215 k	1076 k	—

UDP Flow3

- Does the throughput match the bottleneck throughput of the path?

  Yes, all the flows fit the bandwidth, but Flow3 of TCP has a rate really close to the bandwidth.

- Do you observe the same throughput from TCP and UDP?
  - Similarity: Flow1 and Flow2 have exactly same result in TCP, and so do in UDP.
  - Difference: In the case of TCP, Flow3's rate is almost twice as the others; however, Flow3 of UDP only slightly higher than the others.

- Bonus
  - What have you learned from this lab?
    - The use of mininet
    - The use of Wireshark
    - The method of making directory in python
    - The method of loading files into python
    - The method of parsing pcap files
  - What difficulty have you met in this lab?
    - I am not familiar with nano, which was preinstalled in VB, so I installed vim to solve it.
    - Since I have to use SSH to access Github, I generate SSH keys in VB. Sadly, it went wrong in the beginning, and it takes me a lot of time to solve it.