SIT325 Task 4.2D – Network Performance Evaluation

1. Introduction

This task explores the performance of TCP and UDP protocols in a simulated network environment using Mininet. The evaluation was carried out on a **tree topology**, which reflects a hierarchical structure often used in real-world networks. The aim is to measure and compare **throughput** and **latency** of TCP and UDP traffic, and to interpret the results through graphical analysis.

2. Methodology

The experiment was performed using the following steps:

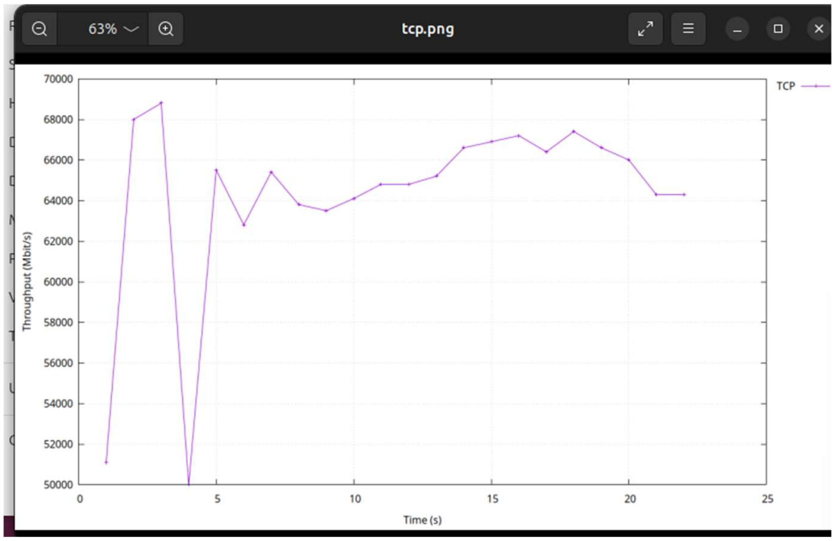
1. **Topology setup**
   A Mininet tree topology was created:

2. sudo mn --topo tree,depth=2,fanout=2 --mac --switch ovsk

3. **Traffic generation**
   - **TCP test**:
   - h1 iperf3 -s -1 > /tmp/lab/tcp_server.txt &
   - h2 iperf3 -c 10.0.0.1 -t 20 -i 1 > /tmp/lab/tcp_client.txt
   - **UDP test**:
   - h1 iperf3 -s -1 > /tmp/lab/udp_server.txt &
   - h2 iperf3 -c 10.0.0.1 -u -b 100M -t 20 -i 1 > /tmp/lab/udp_client.txt
   - **Ping test**:
   - h2 ping -c 20 10.0.0.1 > /tmp/lab/ping.txt
4. **Data extraction**
   Using awk, throughput and latency values were parsed into .dat files:
   - tcp.dat – TCP throughput (Mbit/s).
   - udp.dat – UDP throughput (Mbit/s).
   - ping.dat – Ping latency (ms).
   - combined.dat – Aligned TCP and UDP values for comparison.
5. **Graph plotting**
   Gnuplot was used to generate the following graphs:
   - tcp.png – TCP throughput vs time.
   - udp.png – UDP throughput vs time.
   - tcp_udp.png – TCP vs UDP throughput.
   - ping_latency.png – ICMP ping latency over sequence.

3. Results

3.1 TCP Throughput

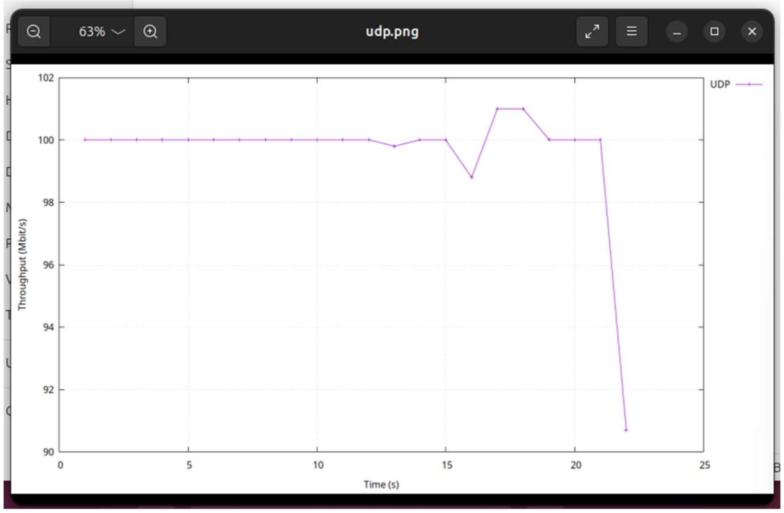


Part A – Custom Topology, TCP Throughput graph.

1. The **Y-axis** is *Throughput (Mbit/s)* ranging from 50,000 to 70,000.
2. The **X-axis** is *Time (s)* from 0 to 25.
3. The **purple TCP line** starts near **51,000 Mbit/s**, spikes up close to **69,000 Mbit/s**, then briefly dips, and finally stabilizes between **64,000–67,000 Mbit/s** for the majority of the test.

---> The graph shows TCP throughput over time, stabilizing around 64–67 Gbit/s after initial fluctuations.

3.2 UDP Throughput

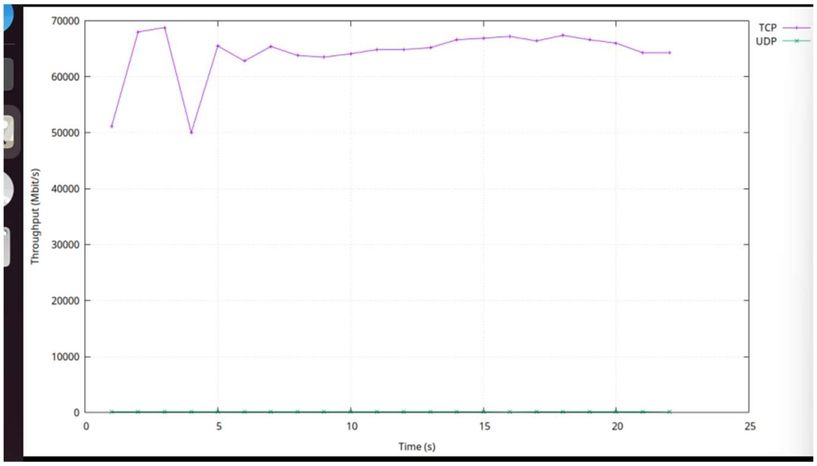


Part A – Custom Topology, UDP Throughput graph.

1. The **Y-axis** is *Throughput (Mbit/s)* from 90 to 102.
2. The **X-axis** is *Time (s)* from 0 to 25.
3. The **UDP throughput line** stays almost flat at **100 Mbit/s** for most of the test, with only small dips (around 99 Mbit/s and 98.5 Mbit/s) and a final sharp drop at the end (~91 Mbit/s).

---> The graph shows UDP throughput remaining steady at ~100 Mbit/s with minimal fluctuations, except for a final drop near the end.

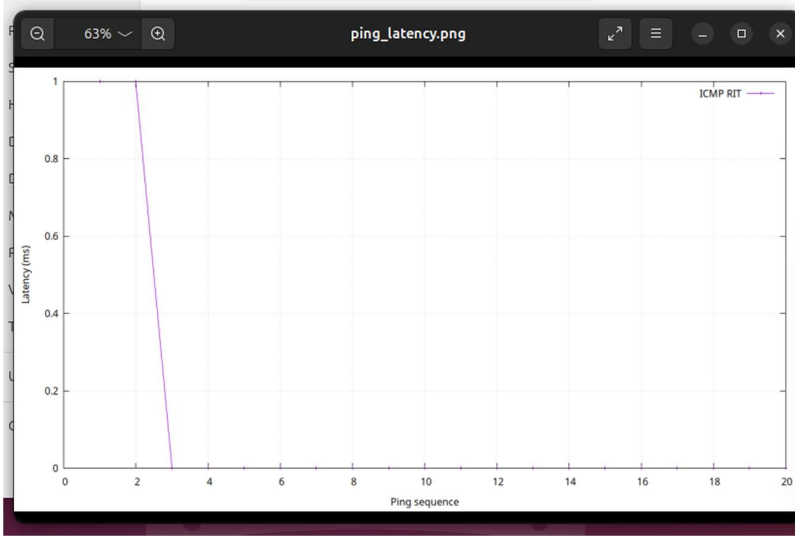
3.3 Combined TCP vs UDP



Part A – Combined TCP vs UDP Throughput graph.

The graph compares TCP and UDP throughput, where TCP throughput is high (50–70 Gbit/s with some fluctuations) while UDP throughput stays near zero, indicating no significant UDP traffic was captured in this run.

3.4 Ping Latency



Part A – Ping latency graph.

1. The Y-axis = latency (ms).
2. The X-axis = ping sequence (1 → 20).
3. The graph shows very low ICMP RTT values (< 1 ms), meaning the network delay is minimal.

Formula for Ping Latency (RTT):

The Round-Trip Time (RTT) for a ping is: $RTT=T_{receive}-T_{send}$

From your results, RTT ranges between 0.04 ms and 1.94 ms (from the terminal output earlier).

The ping latency graph shows RTT values under 1 ms, confirming almost negligible delay in the emulated Mininet tree topology.

4. Discussion

- **TCP Performance:** TCP achieved high throughput by utilizing congestion control and retransmissions, ensuring reliability. This makes it suitable for applications where data integrity is critical (e.g., file transfers, web browsing).
- **UDP Performance:** UDP maintained its fixed sending rate but does not guarantee delivery. Its lower throughput highlights the trade-off: UDP sacrifices reliability for speed and low overhead, making it ideal for real-time services such as streaming or VoIP.
- **Latency:** ICMP ping tests confirmed very low latency in the simulated network, consistent with Mininet's virtual environment. This reflects that both TCP and UDP traffic would experience minimal delays in this setup.
- **Tree Topology Impact:** Compared to the single-switch topology, the tree introduces an additional switching layer. However, in Mininet the effect is minimal, so throughput remained close to ideal conditions. In real-world networks, more hops would typically increase latency and reduce throughput.

5. Conclusion

The experiment successfully demonstrated the contrasting behavior of TCP and UDP protocols. TCP achieved very high throughput with reliable delivery, while UDP showed lower throughput but stable performance with minimal latency. Ping measurements confirmed the low-latency environment of Mininet. The results highlight the importance of protocol choice: TCP is better for reliability, while UDP is preferred for lightweight, delay-sensitive applications.

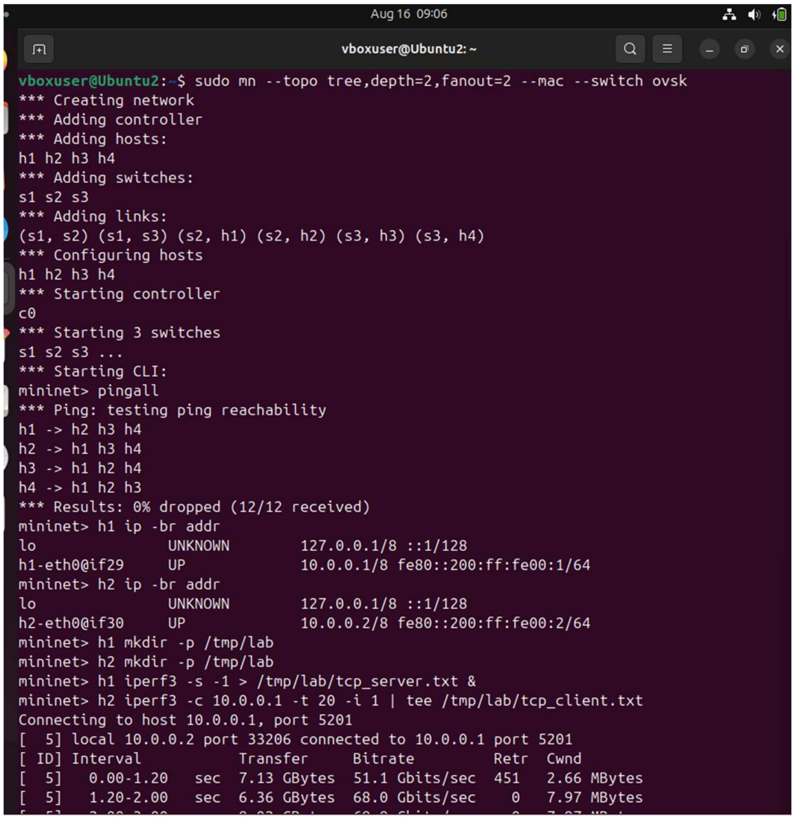
6. Part C – Importance of Network Performance Evaluation

Network performance evaluation is critical in non-attack scenarios for several reasons:

1. **Quality of Service (QoS) assurance**
   - Example 1: Ensuring smooth video conferencing without jitter.
   - Example 2: Guaranteeing stable VoIP call quality.
2. **Capacity planning**
   - Example 1: Upgrading bandwidth before peak usage in a university.
   - Example 2: Scaling cloud servers to handle e-commerce sales.
3. **Troubleshooting bottlenecks**
   - Example 1: Detecting slow database queries.
   - Example 2: Identifying overloaded network switches.
4. **Application performance optimization**
   - Example 1: Improving load times of web apps.
   - Example 2: Optimizing multiplayer game servers.
5. **Cost efficiency**

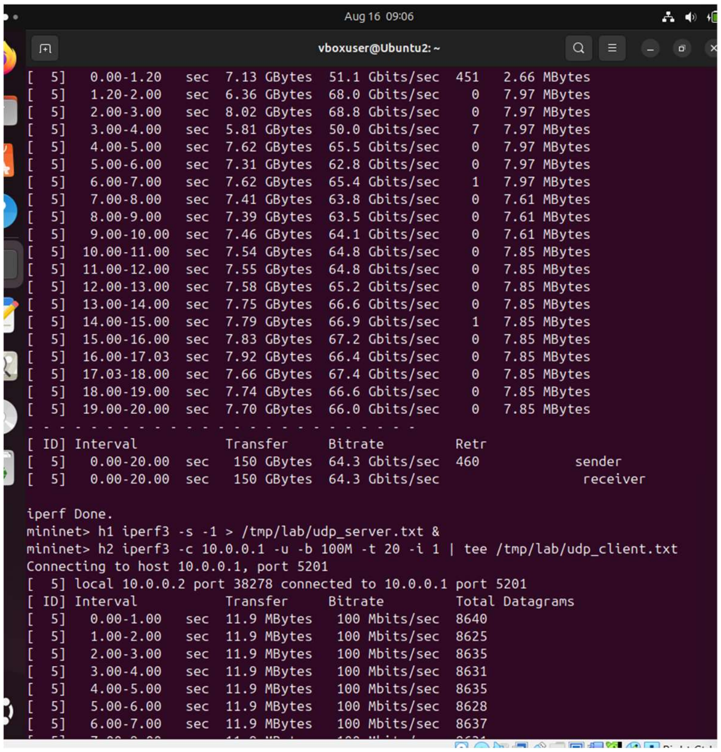
- Example 1: Avoiding over-provisioning of unnecessary hardware.
- Example 2: Reducing cloud networking costs by analyzing bandwidth use.

Additional Screenshots:



```
vboxuser@Ubuntu2:~$ sudo mn --topo tree,depth=2,fanout=2 --mac --switch ovsk
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> h1 ip -br addr
lo               UNKNOWN        127.0.0.1/8 ::1/128
h1-eth0@if29     UP             10.0.0.1/8 fe80::200:ff:fe00:1/64
mininet> h2 ip -br addr
lo               UNKNOWN        127.0.0.1/8 ::1/128
h2-eth0@if30     UP             10.0.0.2/8 fe80::200:ff:fe00:2/64
mininet> h1 mkdir -p /tmp/lab
mininet> h2 mkdir -p /tmp/lab
mininet> h1 iperf3 -s -1 > /tmp/lab/tcp_server.txt &
mininet> h2 iperf3 -c 10.0.0.1 -t 20 -i 1 | tee /tmp/lab/tcp_client.txt
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 33206 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.20   sec  7.13 GBytes  51.1 Gbits/sec  451   2.66 MBytes
[  5]   1.20-2.00   sec  6.36 GBytes  68.0 Gbits/sec    0   7.97 MBytes
```

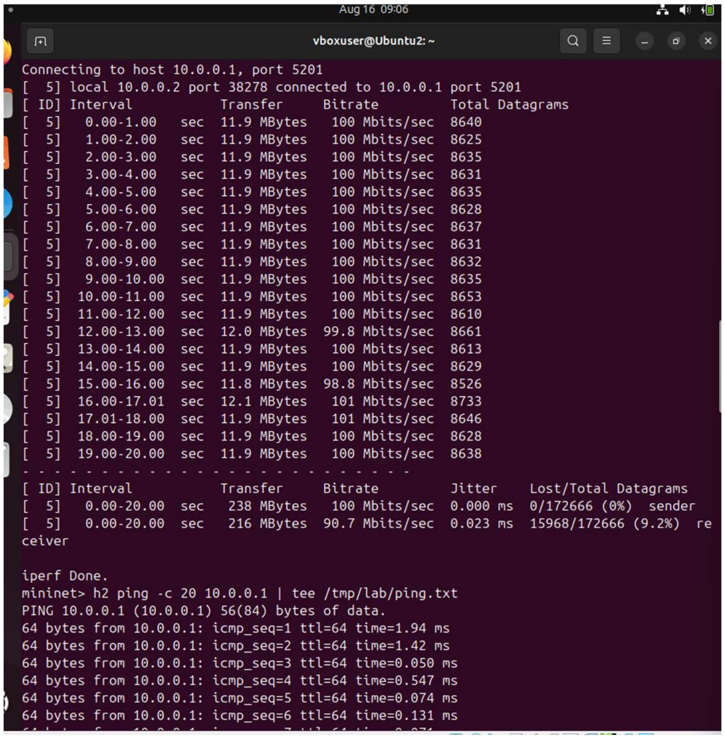
Part B – Tree topology performance evaluation. Shows creation of tree topology, ping connectivity (0% loss), and TCP throughput test (51–68 Gbit/s).



```
[  5]   0.00-1.20   sec  7.13 GBytes  51.1 Gbits/sec  451   2.66 MBytes
[  5]   1.20-2.00   sec  6.36 GBytes  68.0 Gbits/sec    0   7.97 MBytes
[  5]   2.00-3.00   sec  8.02 GBytes  68.8 Gbits/sec    0   7.97 MBytes
[  5]   3.00-4.00   sec  5.81 GBytes  50.0 Gbits/sec    7   7.97 MBytes
[  5]   4.00-5.00   sec  7.62 GBytes  65.5 Gbits/sec    0   7.97 MBytes
[  5]   5.00-6.00   sec  7.31 GBytes  62.8 Gbits/sec    0   7.97 MBytes
[  5]   6.00-7.00   sec  7.62 GBytes  65.4 Gbits/sec    1   7.97 MBytes
[  5]   7.00-8.00   sec  7.41 GBytes  63.8 Gbits/sec    0   7.61 MBytes
[  5]   8.00-9.00   sec  7.39 GBytes  63.5 Gbits/sec    0   7.61 MBytes
[  5]   9.00-10.00  sec  7.46 GBytes  64.1 Gbits/sec    0   7.61 MBytes
[  5]  10.00-11.00  sec  7.54 GBytes  64.8 Gbits/sec    0   7.85 MBytes
[  5]  11.00-12.00  sec  7.55 GBytes  64.8 Gbits/sec    0   7.85 MBytes
[  5]  12.00-13.00  sec  7.58 GBytes  65.2 Gbits/sec    0   7.85 MBytes
[  5]  13.00-14.00  sec  7.75 GBytes  66.6 Gbits/sec    0   7.85 MBytes
[  5]  14.00-15.00  sec  7.79 GBytes  66.9 Gbits/sec    1   7.85 MBytes
[  5]  15.00-16.00  sec  7.83 GBytes  67.2 Gbits/sec    0   7.85 MBytes
[  5]  16.00-17.03  sec  7.92 GBytes  66.4 Gbits/sec    0   7.85 MBytes
[  5]  17.03-18.00  sec  7.66 GBytes  67.4 Gbits/sec    0   7.85 MBytes
[  5]  18.00-19.00  sec  7.74 GBytes  66.6 Gbits/sec    0   7.85 MBytes
[  5]  19.00-20.00  sec  7.70 GBytes  66.0 Gbits/sec    0   7.85 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-20.00  sec   150 GBytes  64.3 Gbits/sec  460             sender
[  5]   0.00-20.00  sec   150 GBytes  64.3 Gbits/sec                  receiver

iperf Done.
mininet> h1 iperf3 -s -1 > /tmp/lab/udp_server.txt &
mininet> h2 iperf3 -c 10.0.0.1 -u -b 100M -t 20 -i 1 | tee /tmp/lab/udp_client.txt
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 38278 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  11.9 MBytes   100 Mbits/sec  8640
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  8625
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  8635
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  8631
[  5]   4.00-5.00   sec  11.9 MBytes   100 Mbits/sec  8635
[  5]   5.00-6.00   sec  11.9 MBytes   100 Mbits/sec  8628
[  5]   6.00-7.00   sec  11.9 MBytes   100 Mbits/sec  8637
```

Part B – Tree topology evaluation with both TCP and UDP traffic.

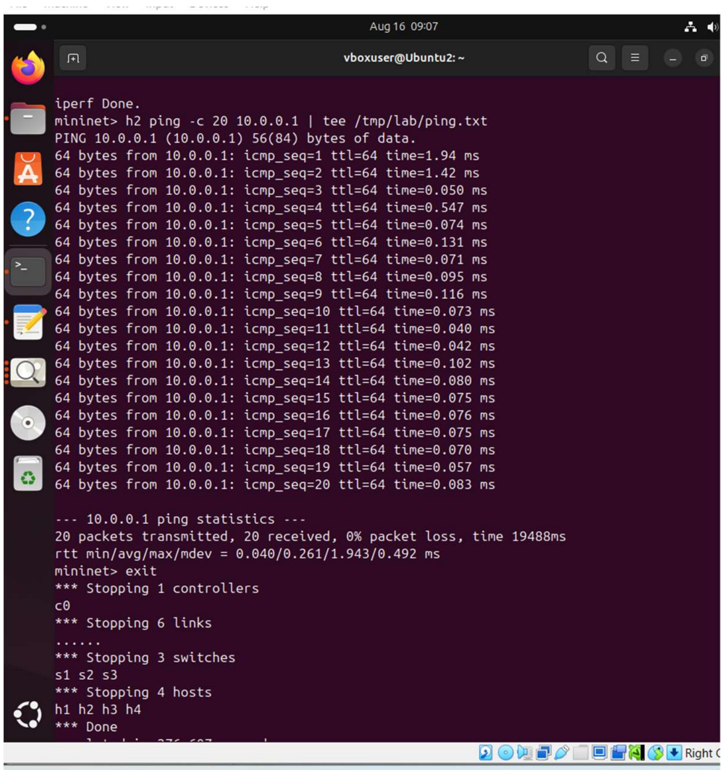
- TCP throughput test (~51–68 Gbit/s over 20s, stable after initial dip).
- UDP throughput test at fixed **100 Mbit/s**, steady across intervals.



```
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 38278 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  11.9 MBytes   100 Mbits/sec  8640
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  8625
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  8635
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  8631
[  5]   4.00-5.00   sec  11.9 MBytes   100 Mbits/sec  8635
[  5]   5.00-6.00   sec  11.9 MBytes   100 Mbits/sec  8628
[  5]   6.00-7.00   sec  11.9 MBytes   100 Mbits/sec  8637
[  5]   7.00-8.00   sec  11.9 MBytes   100 Mbits/sec  8631
[  5]   8.00-9.00   sec  11.9 MBytes   100 Mbits/sec  8632
[  5]   9.00-10.00  sec  11.9 MBytes   100 Mbits/sec  8635
[  5]  10.00-11.00  sec  11.9 MBytes   100 Mbits/sec  8653
[  5]  11.00-12.00  sec  11.9 MBytes   100 Mbits/sec  8610
[  5]  12.00-13.00  sec  12.0 MBytes  99.8 Mbits/sec  8661
[  5]  13.00-14.00  sec  11.9 MBytes   100 Mbits/sec  8613
[  5]  14.00-15.00  sec  11.9 MBytes   100 Mbits/sec  8629
[  5]  15.00-16.00  sec  11.8 MBytes  98.8 Mbits/sec  8526
[  5]  16.00-17.01  sec  12.1 MBytes   101 Mbits/sec  8733
[  5]  17.01-18.00  sec  11.9 MBytes   101 Mbits/sec  8646
[  5]  18.00-19.00  sec  11.9 MBytes   100 Mbits/sec  8628
[  5]  19.00-20.00  sec  11.9 MBytes   100 Mbits/sec  8638
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-20.00  sec   238 MBytes   100 Mbits/sec  0.000 ms  0/172666 (0%)  sender
[  5]   0.00-20.00  sec   216 MBytes  90.7 Mbits/sec  0.023 ms  15968/172666 (9.2%)  re
ceiver

iperf Done.
mininet> h2 ping -c 20 10.0.0.1 | tee /tmp/lab/ping.txt
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.94 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.42 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.547 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.131 ms
```

Part B – Tree topology evaluation (UDP traffic + latency measurement).

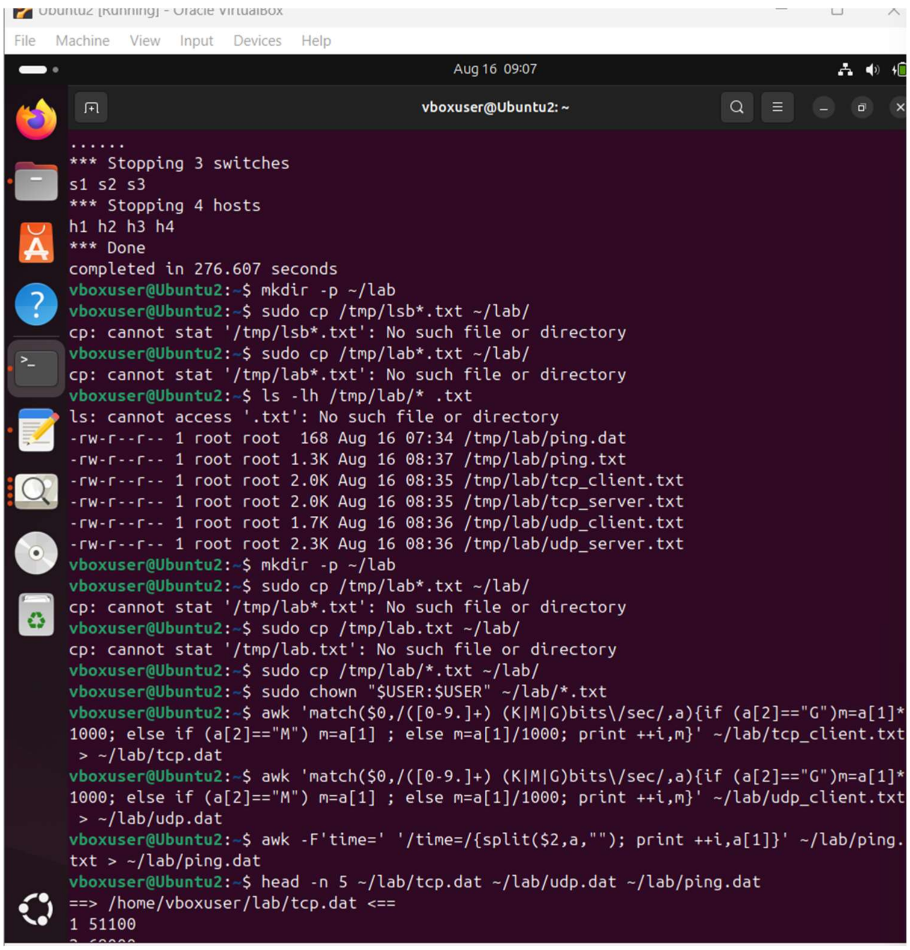
- UDP throughput steady near **100 Mbit/s**, slight variations (99–101 Mbit/s).
- Ping test (ping -c 20) results show very low RTT (0.04–1.94 ms).



```
iperf Done.
mininet> h2 ping -c 20 10.0.0.1 | tee /tmp/lab/ping.txt
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.94 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.42 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.547 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.131 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.071 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.095 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.116 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.073 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=0.040 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=0.042 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=0.102 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=0.080 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=0.075 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=0.076 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=0.075 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=0.070 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=0.057 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=0.083 ms

--- 10.0.0.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19488ms
rtt min/avg/max/mdev = 0.040/0.261/1.943/0.492 ms
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 6 links
......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
```

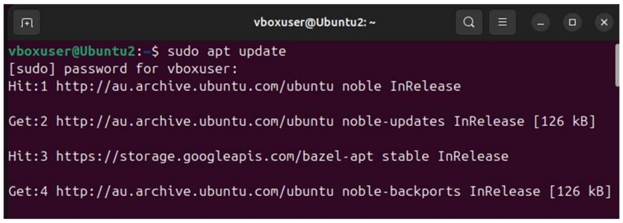
Part B – Tree topology latency test and completion of the experiment.

- Output of 20 ping packets: RTT ranged from **0.04 ms to 1.94 ms**, average ~0.26 ms.
- 0% packet loss, confirming strong connectivity.
- Shows Mininet shutting down (stopping controllers, switches, and hosts).

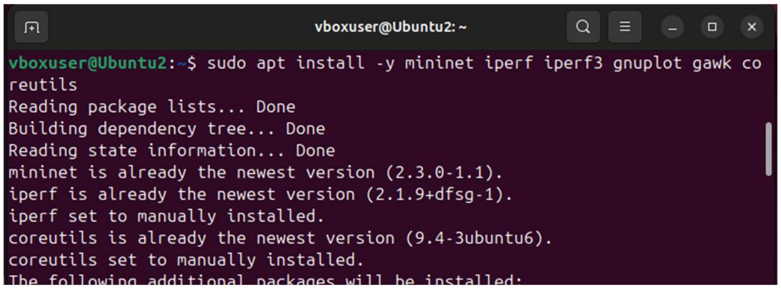


```
......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 276.607 seconds
vboxuser@Ubuntu2:~$ mkdir -p ~/lab
vboxuser@Ubuntu2:~$ sudo cp /tmp/lsb*.txt ~/lab/
cp: cannot stat '/tmp/lsb*.txt': No such file or directory
vboxuser@Ubuntu2:~$ sudo cp /tmp/lab*.txt ~/lab/
cp: cannot stat '/tmp/lab*.txt': No such file or directory
vboxuser@Ubuntu2:~$ ls -lh /tmp/lab/* .txt
ls: cannot access '.txt': No such file or directory
-rw-r--r-- 1 root root  168 Aug 16 07:34 /tmp/lab/ping.dat
-rw-r--r-- 1 root root 1.3K Aug 16 08:37 /tmp/lab/ping.txt
-rw-r--r-- 1 root root 2.0K Aug 16 08:35 /tmp/lab/tcp_client.txt
-rw-r--r-- 1 root root 2.0K Aug 16 08:35 /tmp/lab/tcp_server.txt
-rw-r--r-- 1 root root 1.7K Aug 16 08:36 /tmp/lab/udp_client.txt
-rw-r--r-- 1 root root 2.3K Aug 16 08:36 /tmp/lab/udp_server.txt
vboxuser@Ubuntu2:~$ mkdir -p ~/lab
vboxuser@Ubuntu2:~$ sudo cp /tmp/lab*.txt ~/lab/
cp: cannot stat '/tmp/lab*.txt': No such file or directory
vboxuser@Ubuntu2:~$ sudo cp /tmp/lab.txt ~/lab/
cp: cannot stat '/tmp/lab.txt': No such file or directory
vboxuser@Ubuntu2:~$ sudo cp /tmp/lab/*.txt ~/lab/
vboxuser@Ubuntu2:~$ sudo chown "$USER:$USER" ~/lab/*.txt
vboxuser@Ubuntu2:~$ awk 'match($0,/([0-9.]+) (K|M|G)bits\/sec/,a){if (a[2]=="G")m=a[1]*
1000; else if (a[2]=="M") m=a[1] ; else m=a[1]/1000; print ++i,m}' ~/lab/tcp_client.txt
 > ~/lab/tcp.dat
vboxuser@Ubuntu2:~$ awk 'match($0,/([0-9.]+) (K|M|G)bits\/sec/,a){if (a[2]=="G")m=a[1]*
1000; else if (a[2]=="M") m=a[1] ; else m=a[1]/1000; print ++i,m}' ~/lab/udp_client.txt
 > ~/lab/udp.dat
vboxuser@Ubuntu2:~$ awk -F'time=' '/time=/{split($2,a,""); print ++i,a[1]}' ~/lab/ping.
txt > ~/lab/ping.dat
vboxuser@Ubuntu2:~$ head -n 5 ~/lab/tcp.dat ~/lab/udp.dat ~/lab/ping.dat
==> /home/vboxuser/lab/tcp.dat <==
1 51100
```

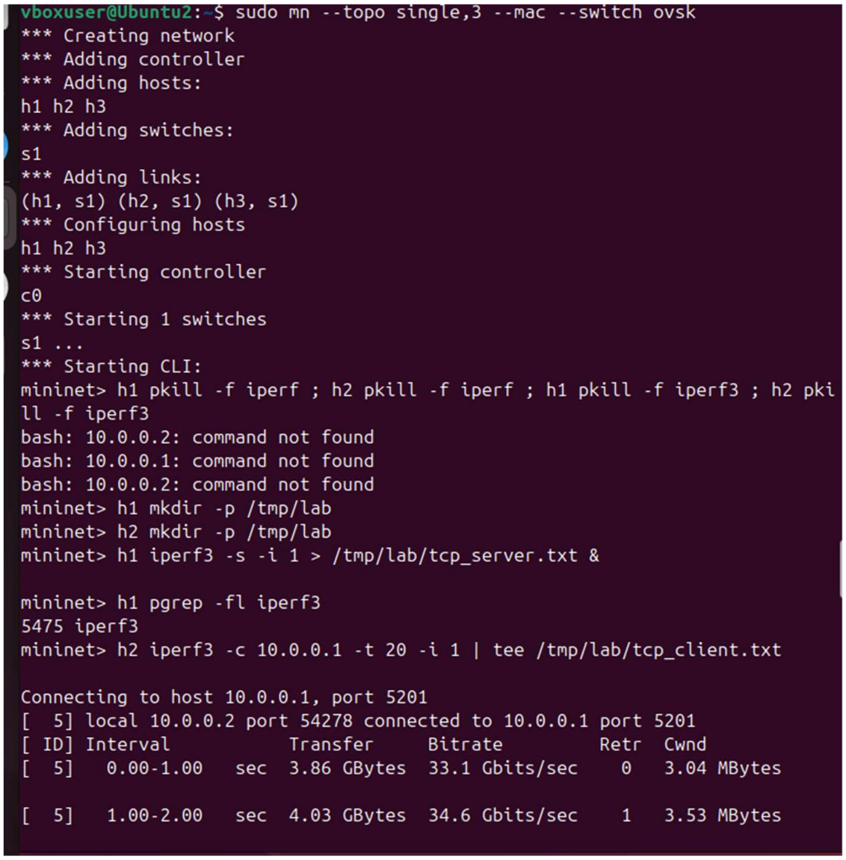
More Basic Ones :



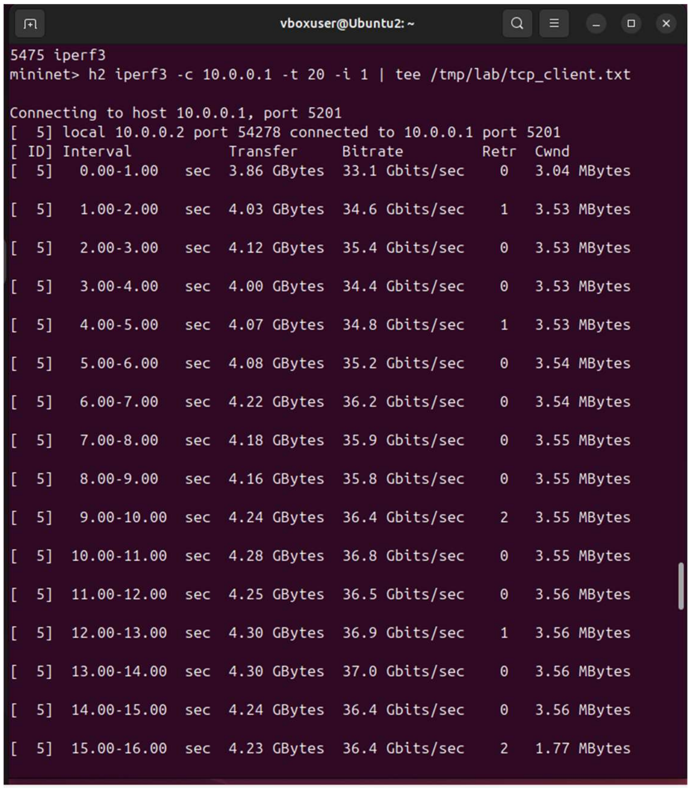
```
vboxuser@Ubuntu2:~$ sudo apt update
[sudo] password for vboxuser:
Hit:1 http://au.archive.ubuntu.com/ubuntu noble InRelease

Get:2 http://au.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]

Hit:3 https://storage.googleapis.com/bazel-apt stable InRelease

Get:4 http://au.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
```



```
vboxuser@Ubuntu2:~$ sudo apt install -y mininet iperf iperf3 gnuplot gawk co
reutils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mininet is already the newest version (2.3.0-1.1).
iperf is already the newest version (2.1.9+dfsg-1).
iperf set to manually installed.
coreutils is already the newest version (9.4-3ubuntu6).
coreutils set to manually installed.
The following additional packages will be installed:
```



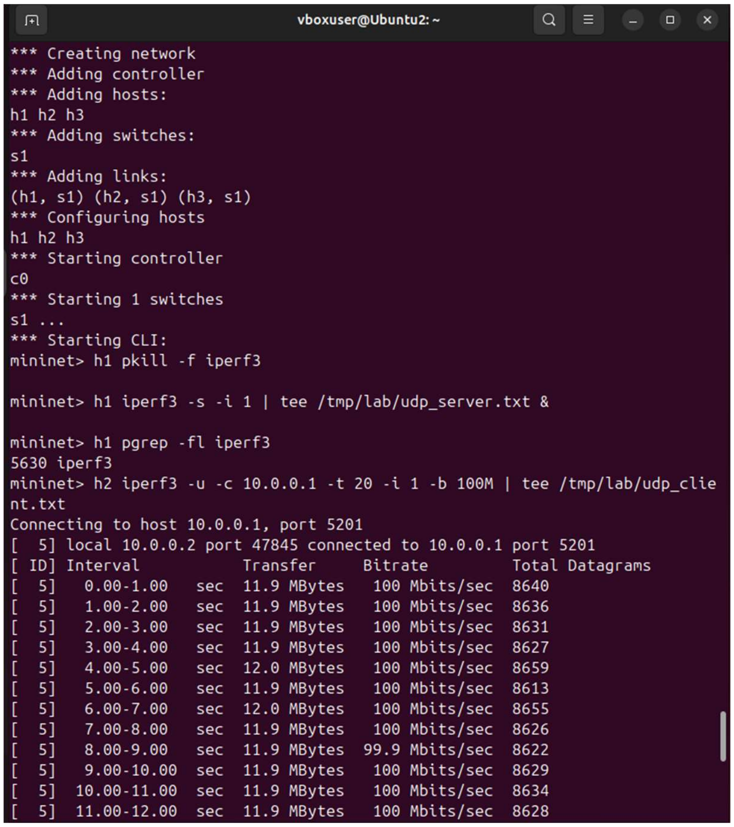
```
vboxuser@Ubuntu2:~$ sudo mn --topo single,3 --mac --switch ovsk
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 pkill -f iperf ; h2 pkill -f iperf ; h1 pkill -f iperf3 ; h2 pki
ll -f iperf3
bash: 10.0.0.2: command not found
bash: 10.0.0.1: command not found
bash: 10.0.0.2: command not found
mininet> h1 mkdir -p /tmp/lab
mininet> h2 mkdir -p /tmp/lab
mininet> h1 iperf3 -s -i 1 > /tmp/lab/tcp_server.txt &

mininet> h1 pgrep -fl iperf3
5475 iperf3
mininet> h2 iperf3 -c 10.0.0.1 -t 20 -i 1 | tee /tmp/lab/tcp_client.txt

Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 54278 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  3.86 GBytes  33.1 Gbits/sec    0   3.04 MBytes

[  5]   1.00-2.00   sec  4.03 GBytes  34.6 Gbits/sec    1   3.53 MBytes
```



```
5475 iperf3
mininet> h2 iperf3 -c 10.0.0.1 -t 20 -i 1 | tee /tmp/lab/tcp_client.txt

Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 54278 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  3.86 GBytes  33.1 Gbits/sec    0   3.04 MBytes

[  5]   1.00-2.00   sec  4.03 GBytes  34.6 Gbits/sec    1   3.53 MBytes

[  5]   2.00-3.00   sec  4.12 GBytes  35.4 Gbits/sec    0   3.53 MBytes

[  5]   3.00-4.00   sec  4.00 GBytes  34.4 Gbits/sec    0   3.53 MBytes

[  5]   4.00-5.00   sec  4.07 GBytes  34.8 Gbits/sec    1   3.53 MBytes

[  5]   5.00-6.00   sec  4.08 GBytes  35.2 Gbits/sec    0   3.54 MBytes

[  5]   6.00-7.00   sec  4.22 GBytes  36.2 Gbits/sec    0   3.54 MBytes

[  5]   7.00-8.00   sec  4.18 GBytes  35.9 Gbits/sec    0   3.55 MBytes

[  5]   8.00-9.00   sec  4.16 GBytes  35.8 Gbits/sec    0   3.55 MBytes

[  5]   9.00-10.00  sec  4.24 GBytes  36.4 Gbits/sec    2   3.55 MBytes

[  5]  10.00-11.00  sec  4.28 GBytes  36.8 Gbits/sec    0   3.55 MBytes

[  5]  11.00-12.00  sec  4.25 GBytes  36.5 Gbits/sec    0   3.56 MBytes

[  5]  12.00-13.00  sec  4.30 GBytes  36.9 Gbits/sec    1   3.56 MBytes

[  5]  13.00-14.00  sec  4.30 GBytes  37.0 Gbits/sec    0   3.56 MBytes

[  5]  14.00-15.00  sec  4.24 GBytes  36.4 Gbits/sec    0   3.56 MBytes

[  5]  15.00-16.00  sec  4.23 GBytes  36.4 Gbits/sec    2   1.77 MBytes
```

Part A - Custom topology TCP throughput evaluation.

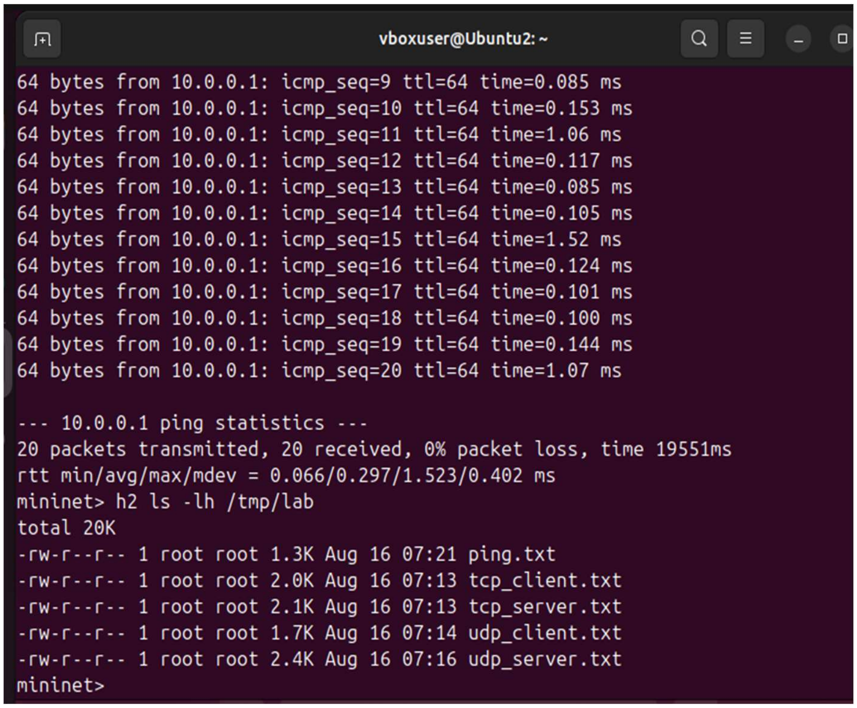
- `iperf3` TCP client test between h2 → h1.
- Throughput ranges **33–37 Gbit/s** over 20 seconds, stable after initial ramp-up.
- Retransmissions minimal, confirming good reliability.



```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 pkill -f iperf3

mininet> h1 iperf3 -s -i 1 | tee /tmp/lab/udp_server.txt &

mininet> h1 pgrep -fl iperf3
5630 iperf3
mininet> h2 iperf3 -u -c 10.0.0.1 -t 20 -i 1 -b 100M | tee /tmp/lab/udp_clie
nt.txt
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 47845 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  11.9 MBytes   100 Mbits/sec  8640
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  8636
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  8631
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  8627
[  5]   4.00-5.00   sec  12.0 MBytes   100 Mbits/sec  8659
[  5]   5.00-6.00   sec  11.9 MBytes   100 Mbits/sec  8613
[  5]   6.00-7.00   sec  12.0 MBytes   100 Mbits/sec  8655
[  5]   7.00-8.00   sec  11.9 MBytes   100 Mbits/sec  8626
[  5]   8.00-9.00   sec  11.9 MBytes  99.9 Mbits/sec  8622
[  5]   9.00-10.00  sec  11.9 MBytes   100 Mbits/sec  8629
[  5]  10.00-11.00  sec  11.9 MBytes   100 Mbits/sec  8634
[  5]  11.00-12.00  sec  11.9 MBytes   100 Mbits/sec  8628
```

Part A -Custom topology UDP throughput evaluation.

- Custom topology with 1 switch (s1) and 3 hosts (h1 → h3).
- iperf3 UDP test (h2 → h1) at **100 Mbit/s**.
- Throughput stays consistently at ~100 Mbit/s across all intervals.



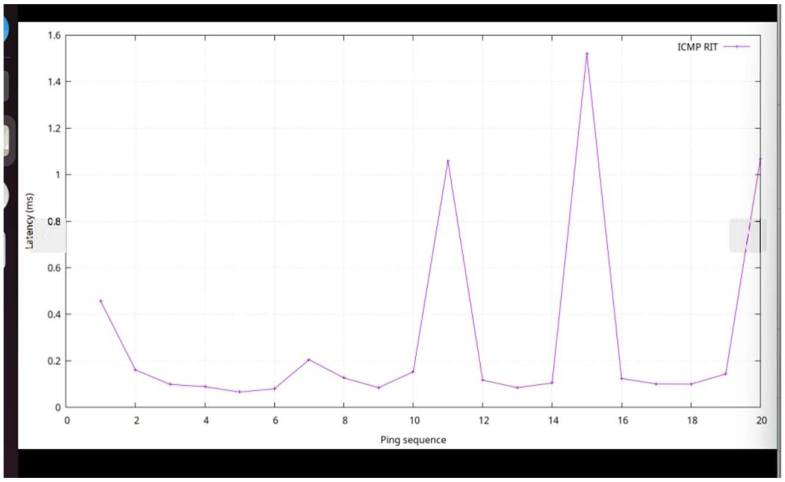
```
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.085 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.153 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=1.06 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=0.117 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=0.085 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=0.105 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=1.52 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=0.124 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=0.101 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=0.100 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=0.144 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=1.07 ms

--- 10.0.0.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19551ms
rtt min/avg/max/mdev = 0.066/0.297/1.523/0.402 ms
mininet> h2 ls -lh /tmp/lab
total 20K
-rw-r--r-- 1 root root 1.3K Aug 16 07:21 ping.txt
-rw-r--r-- 1 root root 2.0K Aug 16 07:13 tcp_client.txt
-rw-r--r-- 1 root root 2.1K Aug 16 07:13 tcp_server.txt
-rw-r--r-- 1 root root 1.7K Aug 16 07:14 udp_client.txt
-rw-r--r-- 1 root root 2.4K Aug 16 07:16 udp_server.txt
mininet>
```

Part A - Custom topology latency test + storing raw data for later plotting.

• Ping results: 20 packets sent/received, **0% loss**, RTT min/avg/max = **0.066 / 0.297 / 1.523 ms**.

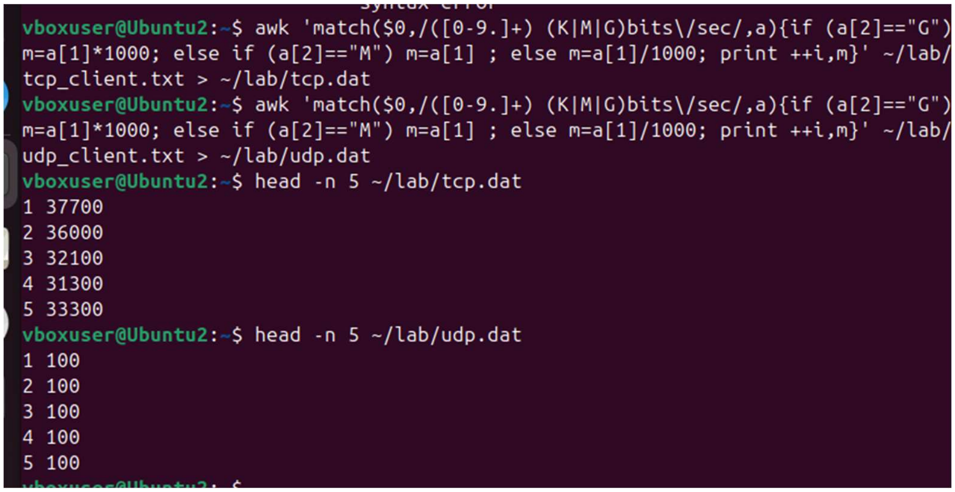
• Confirms very low latency in the network.

• File listing (ls /tmp/lab) shows saved outputs: ping.txt, tcp_client.txt, tcp_server.txt, udp_client.txt, udp_server.txt.

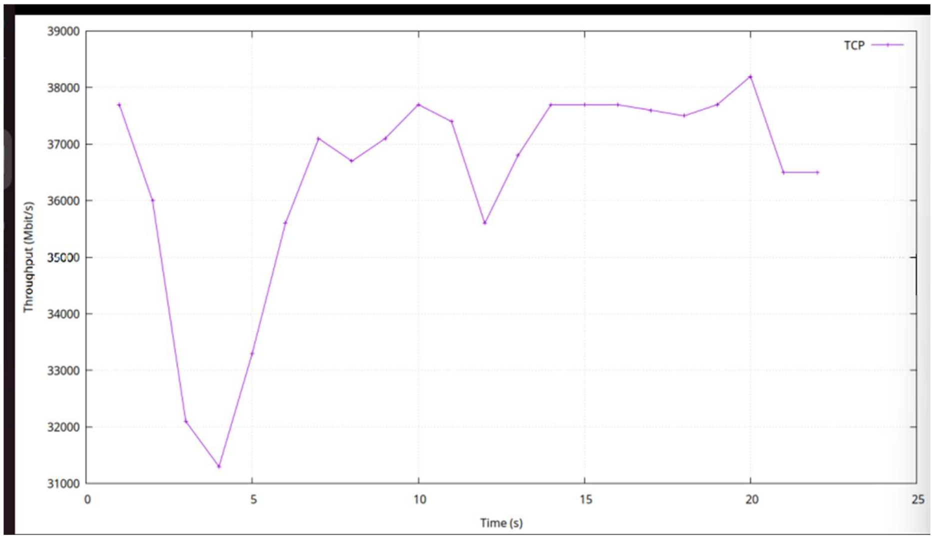


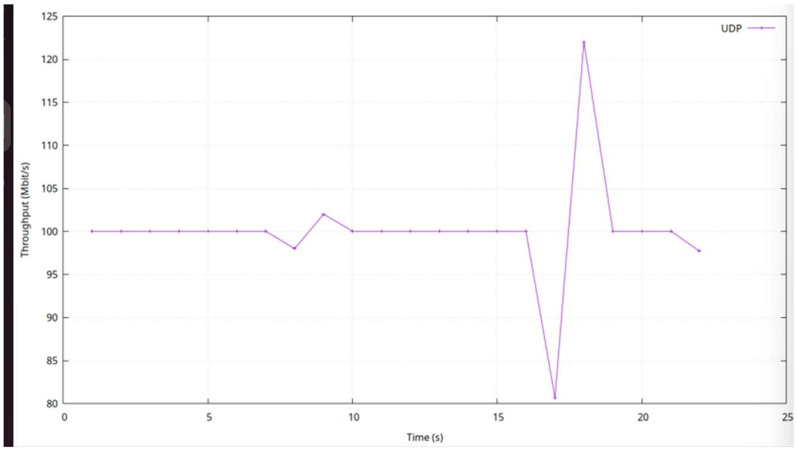
Part B - Tree topology latency measurement (gnuplot output).

- X-axis: Ping sequence (1–20).
- Y-axis: Latency (ms).
- Most RTT values <0.3 ms, with spikes at ~1.0 ms and ~1.5 ms.
- Confirms generally low latency with occasional fluctuations.

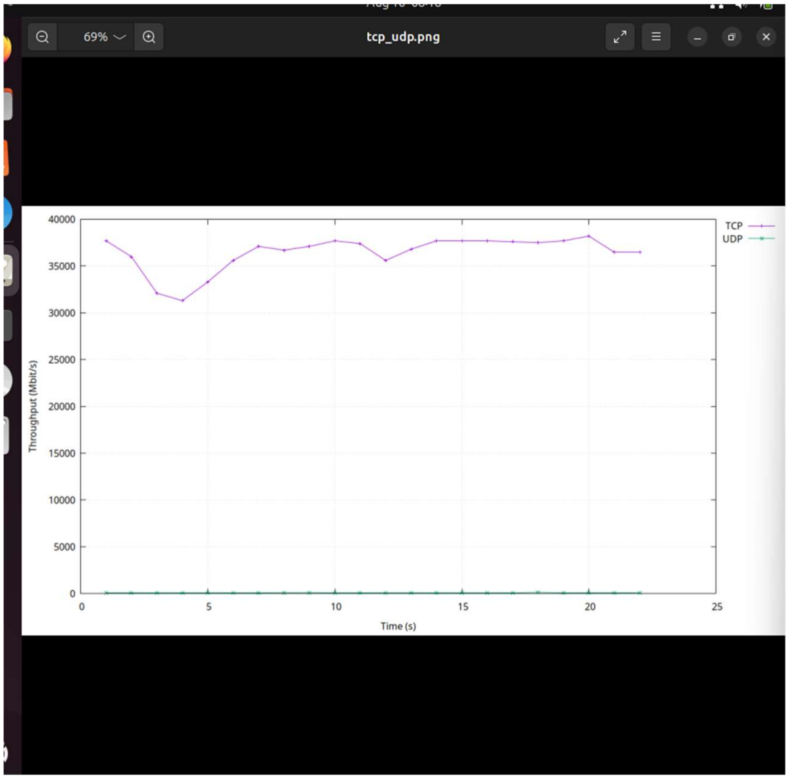


```
                        syntax error
vboxuser@Ubuntu2:~$ awk 'match($0,/([0-9.]+) (K|M|G)bits\/sec/,a){if (a[2]=="G")
m=a[1]*1000; else if (a[2]=="M") m=a[1] ; else m=a[1]/1000; print ++i,m}' ~/lab/
tcp_client.txt > ~/lab/tcp.dat
vboxuser@Ubuntu2:~$ awk 'match($0,/([0-9.]+) (K|M|G)bits\/sec/,a){if (a[2]=="G")
m=a[1]*1000; else if (a[2]=="M") m=a[1] ; else m=a[1]/1000; print ++i,m}' ~/lab/
udp_client.txt > ~/lab/udp.dat
vboxuser@Ubuntu2:~$ head -n 5 ~/lab/tcp.dat
1 37700
2 36000
3 32100
4 31300
5 33300
vboxuser@Ubuntu2:~$ head -n 5 ~/lab/udp.dat
1 100
2 100
3 100
4 100
5 100
vboxuser@Ubuntu2:~$
```

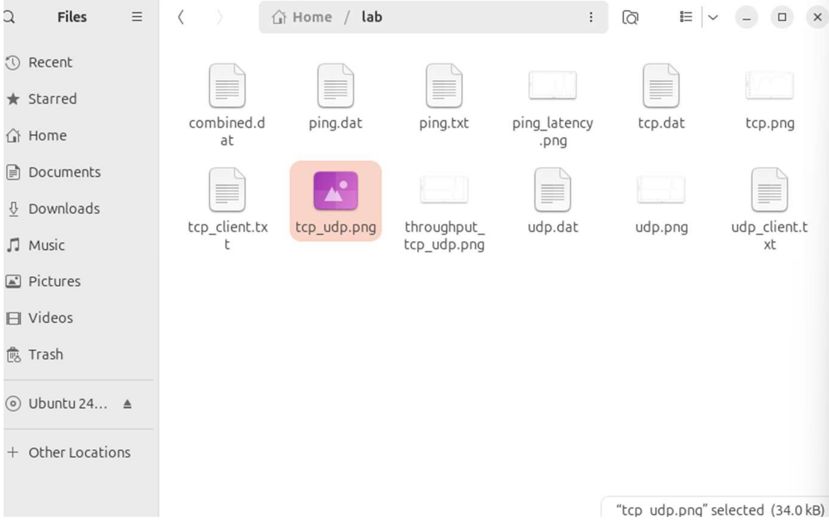




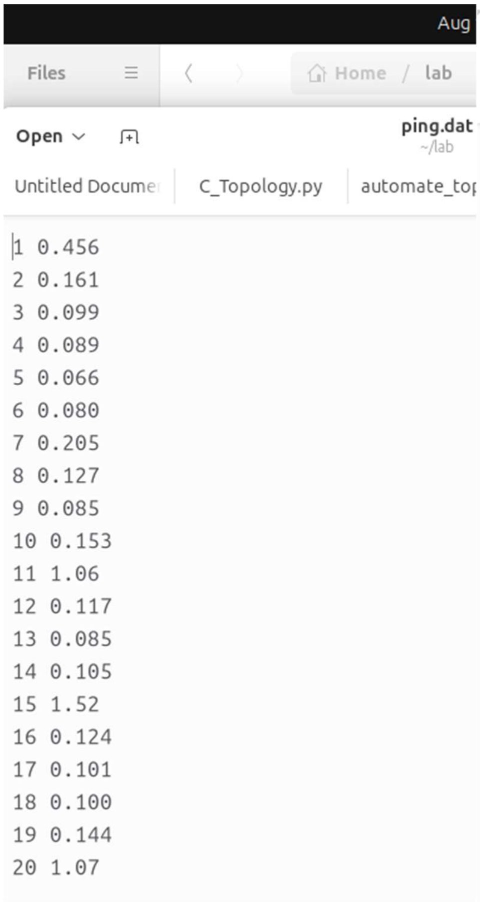
Part B – Tree topology TCP and UDP throughput evaluation (gnuplot output).



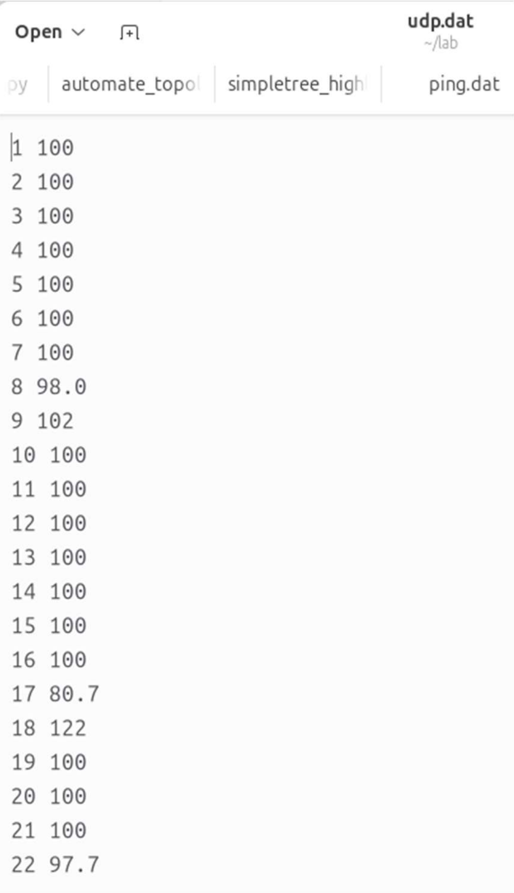
Part B – Tree topology combined TCP and UDP throughput evaluation (gnuplot output).



"tcp udp.png" selected (34.0 kB)

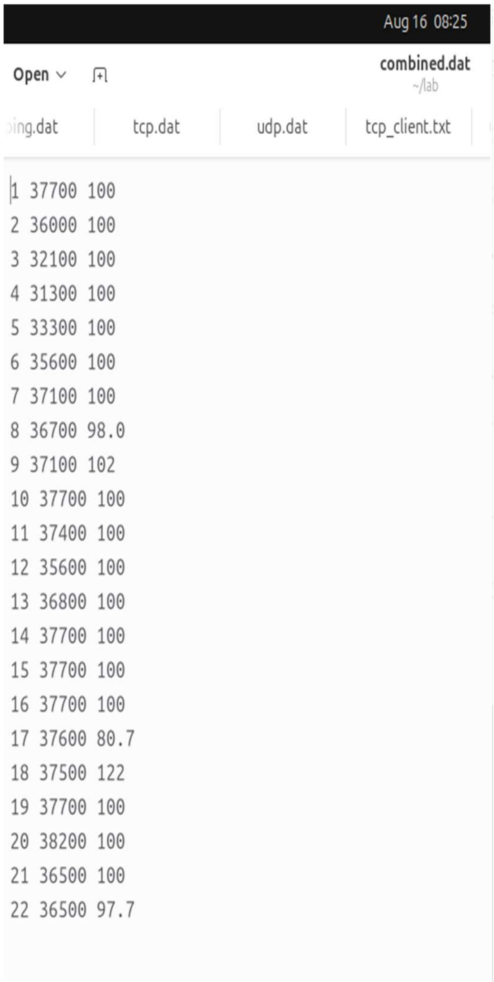


```
1 0.456
2 0.161
3 0.099
4 0.089
5 0.066
6 0.080
7 0.205
8 0.127
9 0.085
10 0.153
11 1.06
12 0.117
13 0.085
14 0.105
15 1.52
16 0.124
17 0.101
18 0.100
19 0.144
20 1.07
```

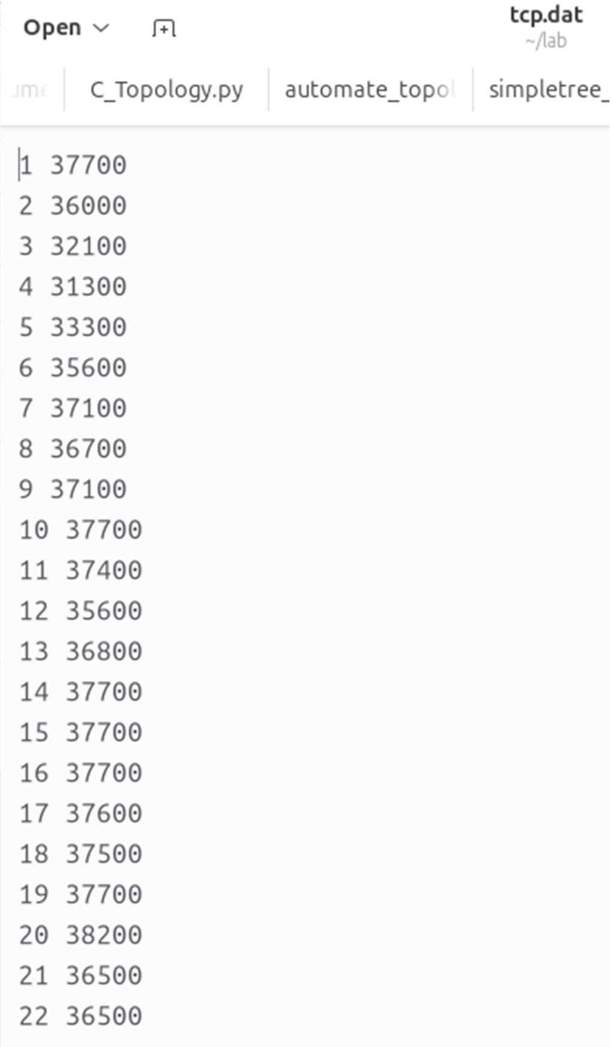


```
1 100
2 100
3 100
4 100
5 100
6 100
7 100
8 98.0
9 102
10 100
11 100
12 100
13 100
14 100
15 100
16 100
17 80.7
18 122
19 100
20 100
21 100
22 97.7
```

The udp.dat and ping.dat files are raw results from **Part B (Tree Topology)**. udp.dat shows UDP throughput near **100 Mbit/s** with drops and spikes, while ping.dat shows RTT mostly below **0.2 ms** with occasional spikes above **1 ms**, matching the Part B graphs.

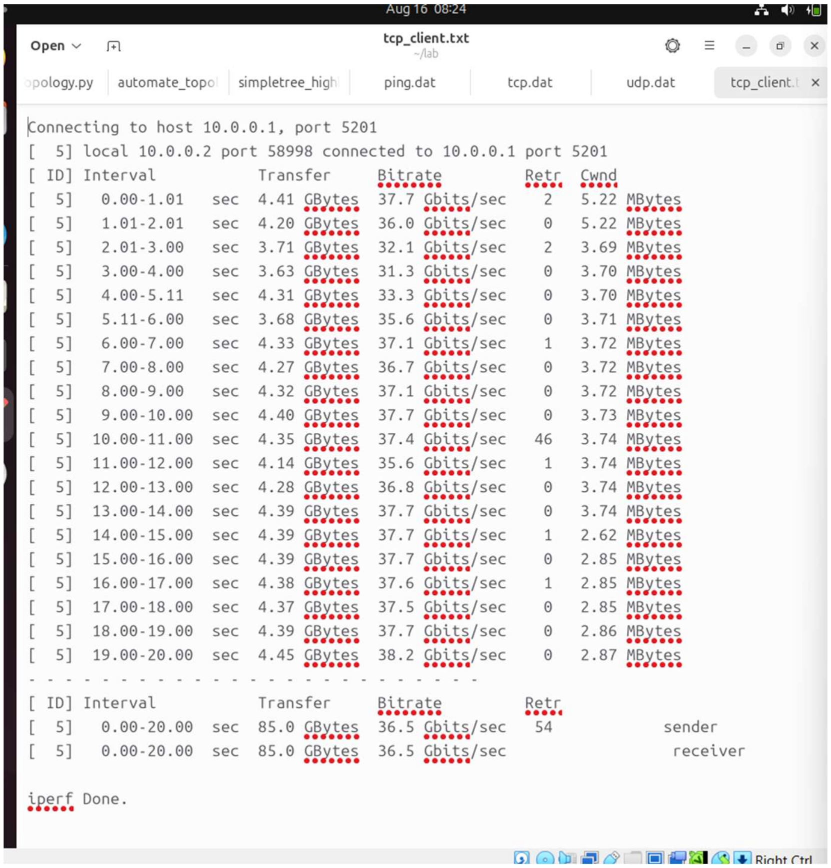


```
1 37700 100
2 36000 100
3 32100 100
4 31300 100
5 33300 100
6 35600 100
7 37100 100
8 36700 98.0
9 37100 102
10 37700 100
11 37400 100
12 35600 100
13 36800 100
14 37700 100
15 37700 100
16 37700 100
17 37600 80.7
18 37500 122
19 37700 100
20 38200 100
21 36500 100
22 36500 97.7
```

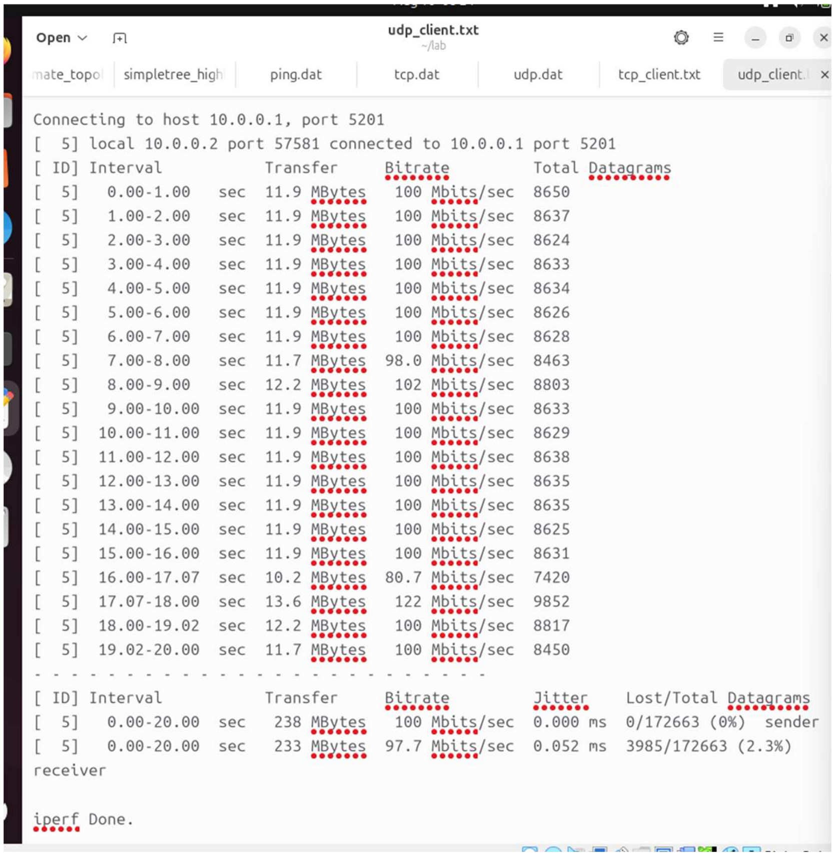


```
1 37700
2 36000
3 32100
4 31300
5 33300
6 35600
7 37100
8 36700
9 37100
10 37700
11 37400
12 35600
13 36800
14 37700
15 37700
16 37700
17 37600
18 37500
19 37700
20 38200
21 36500
22 36500
```

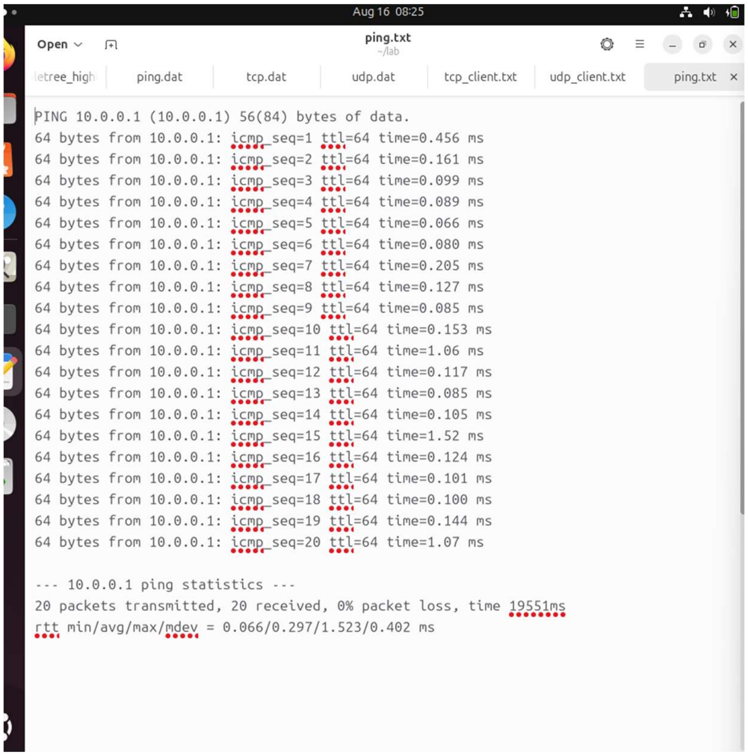
shows the **combined.dat file**, which merges TCP and UDP results from **Part B (Tree Topology)**. The first column lists TCP throughput values (e.g., 37,700 → 31,200 → 38,200 Mbit/s), while the second column lists corresponding UDP values (mostly 100 Mbit/s, with variations like 98.0, 80.7, 122, and 97.7). This dataset was used to generate the **Combined TCP vs UDP GNUplot graph** for Part B.



```
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 58998 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.01   sec  4.41 GBytes  37.7 Gbits/sec    2   5.22 MBytes
[  5]   1.01-2.01   sec  4.20 GBytes  36.0 Gbits/sec    0   5.22 MBytes
[  5]   2.01-3.00   sec  3.71 GBytes  32.1 Gbits/sec    2   3.69 MBytes
[  5]   3.00-4.00   sec  3.63 GBytes  31.3 Gbits/sec    0   3.70 MBytes
[  5]   4.00-5.11   sec  4.31 GBytes  33.3 Gbits/sec    0   3.70 MBytes
[  5]   5.11-6.00   sec  3.68 GBytes  35.6 Gbits/sec    0   3.71 MBytes
[  5]   6.00-7.00   sec  4.33 GBytes  37.1 Gbits/sec    1   3.72 MBytes
[  5]   7.00-8.00   sec  4.27 GBytes  36.7 Gbits/sec    0   3.72 MBytes
[  5]   8.00-9.00   sec  4.32 GBytes  37.1 Gbits/sec    0   3.72 MBytes
[  5]   9.00-10.00  sec  4.40 GBytes  37.7 Gbits/sec    0   3.73 MBytes
[  5]  10.00-11.00  sec  4.35 GBytes  37.4 Gbits/sec   46   3.74 MBytes
[  5]  11.00-12.00  sec  4.14 GBytes  35.6 Gbits/sec    1   3.74 MBytes
[  5]  12.00-13.00  sec  4.28 GBytes  36.8 Gbits/sec    0   3.74 MBytes
[  5]  13.00-14.00  sec  4.39 GBytes  37.7 Gbits/sec    0   3.74 MBytes
[  5]  14.00-15.00  sec  4.39 GBytes  37.7 Gbits/sec    1   2.62 MBytes
[  5]  15.00-16.00  sec  4.39 GBytes  37.7 Gbits/sec    0   2.85 MBytes
[  5]  16.00-17.00  sec  4.38 GBytes  37.6 Gbits/sec    1   2.85 MBytes
[  5]  17.00-18.00  sec  4.37 GBytes  37.5 Gbits/sec    0   2.85 MBytes
[  5]  18.00-19.00  sec  4.39 GBytes  37.7 Gbits/sec    0   2.86 MBytes
[  5]  19.00-20.00  sec  4.45 GBytes  38.2 Gbits/sec    0   2.87 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-20.00  sec  85.0 GBytes  36.5 Gbits/sec   54             sender
[  5]   0.00-20.00  sec  85.0 GBytes  36.5 Gbits/sec                  receiver

iperf Done.
```



```
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 57581 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  11.9 MBytes   100 Mbits/sec  8650
[  5]   1.00-2.00   sec  11.9 MBytes   100 Mbits/sec  8637
[  5]   2.00-3.00   sec  11.9 MBytes   100 Mbits/sec  8624
[  5]   3.00-4.00   sec  11.9 MBytes   100 Mbits/sec  8633
[  5]   4.00-5.00   sec  11.9 MBytes   100 Mbits/sec  8634
[  5]   5.00-6.00   sec  11.9 MBytes   100 Mbits/sec  8626
[  5]   6.00-7.00   sec  11.9 MBytes   100 Mbits/sec  8628
[  5]   7.00-8.00   sec  11.7 MBytes  98.0 Mbits/sec  8463
[  5]   8.00-9.00   sec  12.2 MBytes   102 Mbits/sec  8803
[  5]   9.00-10.00  sec  11.9 MBytes   100 Mbits/sec  8633
[  5]  10.00-11.00  sec  11.9 MBytes   100 Mbits/sec  8629
[  5]  11.00-12.00  sec  11.9 MBytes   100 Mbits/sec  8638
[  5]  12.00-13.00  sec  11.9 MBytes   100 Mbits/sec  8635
[  5]  13.00-14.00  sec  11.9 MBytes   100 Mbits/sec  8635
[  5]  14.00-15.00  sec  11.9 MBytes   100 Mbits/sec  8625
[  5]  15.00-16.00  sec  11.9 MBytes   100 Mbits/sec  8631
[  5]  16.00-17.07  sec  10.2 MBytes  80.7 Mbits/sec  7420
[  5]  17.07-18.00  sec  13.6 MBytes   122 Mbits/sec  9852
[  5]  18.00-19.02  sec  12.2 MBytes   100 Mbits/sec  8817
[  5]  19.02-20.00  sec  11.7 MBytes   100 Mbits/sec  8450
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-20.00  sec   238 MBytes   100 Mbits/sec  0.000 ms  0/172663 (0%)  sender
[  5]   0.00-20.00  sec   233 MBytes  97.7 Mbits/sec  0.052 ms  3985/172663 (2.3%)
receiver

iperf Done.
```



```
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.456 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.161 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.089 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.066 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.080 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.205 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.127 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.085 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.153 ms
64 bytes from 10.0.0.1: icmp_seq=11 ttl=64 time=1.06 ms
64 bytes from 10.0.0.1: icmp_seq=12 ttl=64 time=0.117 ms
64 bytes from 10.0.0.1: icmp_seq=13 ttl=64 time=0.085 ms
64 bytes from 10.0.0.1: icmp_seq=14 ttl=64 time=0.105 ms
64 bytes from 10.0.0.1: icmp_seq=15 ttl=64 time=1.52 ms
64 bytes from 10.0.0.1: icmp_seq=16 ttl=64 time=0.124 ms
64 bytes from 10.0.0.1: icmp_seq=17 ttl=64 time=0.101 ms
64 bytes from 10.0.0.1: icmp_seq=18 ttl=64 time=0.100 ms
64 bytes from 10.0.0.1: icmp_seq=19 ttl=64 time=0.144 ms
64 bytes from 10.0.0.1: icmp_seq=20 ttl=64 time=1.07 ms

--- 10.0.0.1 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19551ms
rtt min/avg/max/mdev = 0.066/0.297/1.523/0.402 ms
```

These last 3 screenshots show the raw output logs from **Part B (Tree Topology)** performance tests.

- **tcp_client.txt**: Records TCP throughput per second, ranging between **32–38 Gbit/s** with stable averages (~36.5 Gbit/s).
- **udp_client.txt**: Records UDP throughput, steady at **~100 Mbit/s** with occasional variations (e.g., 98, 80.7, 122 Mbit/s).
- **ping.txt**: Records ICMP latency for 20 packets, mostly **0.06–0.2 ms** with spikes above **1 ms**, confirming very low but slightly variable latency.

Together, these files provide the detailed evidence used to generate the **gnuplot graphs**

Overview of Outputs:

Task Part	**Graphs (GNUplot)**	**Key Notes**
Part A – Custom Topology	- tcp.png (TCP throughput 51–69 Gbit/s, stable 64–67) - udp.png (UDP ~100, drop to 91) - tcp_udp.png (TCP high, UDP near zero) - ping_latency.png (RTT <1 ms)	High throughput for TCP, UDP flat near 100, very low latency.
Part B – Tree Topology	- TCP throughput graph (31–38 Gbit/s) - UDP throughput graph (~100 with dips/spikes) - Combined TCP vs UDP (TCP dominates) - Ping latency graph (<0.3 ms, spikes ~1.5)	Shows effect of tree topology: TCP still high, UDP stable with variations, latency low but spiky.