Statement and Confirmation of Own Work

A signed copy of this form must be submitted with every assignment. If the statement is missing your work may not be marked.

Student Declaration

I confirm the following details:

Student Name:	Johnson Kenisha Corera
Student ID Number:	c23020001@cicra.edu.lk
Qualification:	Bachelor in Cyber Security
Unit:	SIT325 Advanced Network Security
Centre:	CICRA Campus
Word Count:	1438

I have read and understood both *Deakin Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.

I confirm that I have not exceeded the stipulated word limit by more than 10%.

I confirm that this is my own work and that I have not colluded or plagiarized any part of it.

Candidate Signature:	J. Kw.
Date:	03/12/2024

Task 4.2 D

Table of Contents

Table of Figures

Table of Acronyms

TCP	Transmission Control Protocol
SLA	Service Level Agreement
MSP	Managed Service Provider
UDP	User Datagram Protocol
AWS	Amazon Web Services
B2B	Business-to-Business
WAN	Wide Area Network
RTT	Round Trip Time
SD	Standard Deviations
FPVD	Force Protection Video Equipment Corp

Introduction

- In this report, I provide an overview of my performance on Task 4.2D of the SIT325: Advanced Network Security course. The objective was to capture the raw network performance in the customized small-scale WAN topology and then in front of actual experiment create traffic profile using Iperf and then plot it with Gnuplot.

Part A

- As usual, the first part of accomplishing Task 4.1P involved exercising performance measurement, with a focus on TCP traffic within the network. I conducted a second UDP traffic analysis to complete the Task 4.2D. I also decided to use application called Iperf to measure the, this test measures the throughput on both TCP and UDP protocols. The collected results were stored for later use.

- I therefore later opened two terminals for hosts 1 and 2 using ‘xterm ‘following the creation of the Mininet topology.

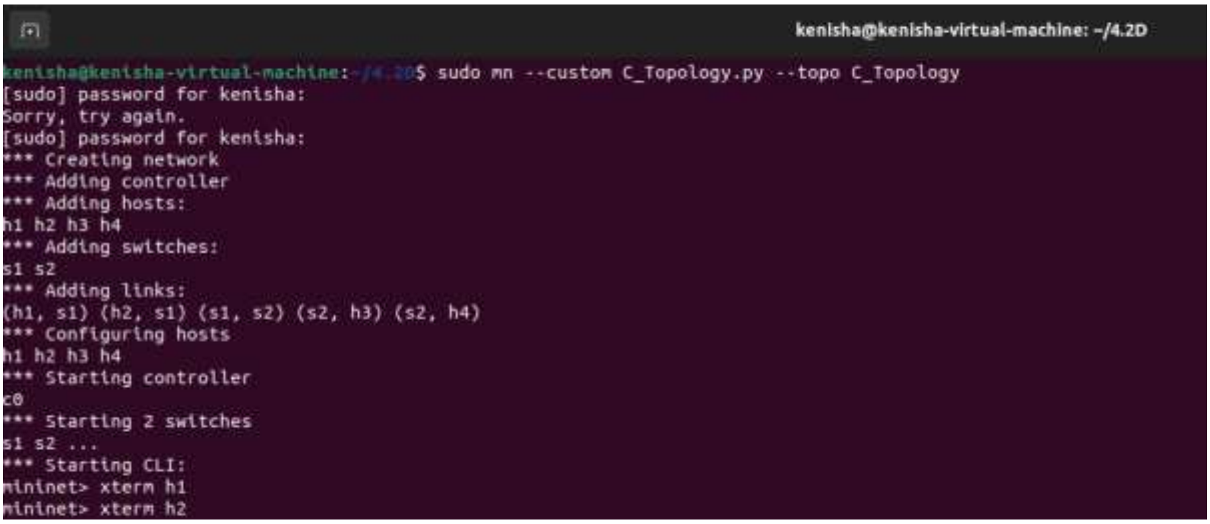


Figure 1: Start Mininet

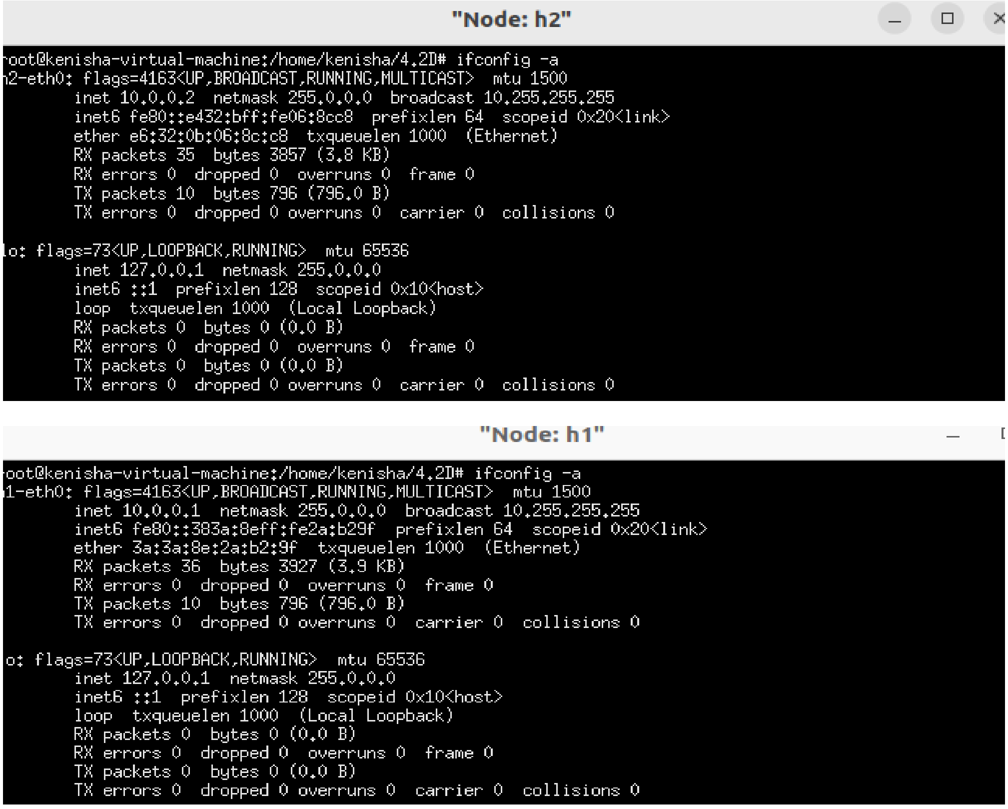


Figure 2: used xterm for h1 and h2

- After that I joined Host 1 as a server and Host 2 as a client. I started the UDP connection, beginning with TCP connection.

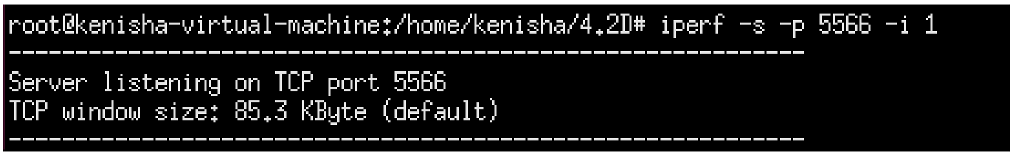


Figure 3: To receive tcp connection I started h1

- I then put the server TCP packets through the transmission and documented the results in tcp_result.txt.

```
root@kenisha-virtual-machine:/home/kenisha/4.2D# iperf -s -p 5566 -i 1
------------------------------------------------------------
Server listening on TCP port 5566
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 5566 connected with 10.0.0.2 port 52530
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  1.51 GBytes  13.0 Gbits/sec
[  1] 1.0000-2.0000 sec  2.10 GBytes  18.1 Gbits/sec
[  1] 2.0000-3.0000 sec  2.18 GBytes  18.8 Gbits/sec
[  1] 3.0000-4.0000 sec  2.08 GBytes  17.9 Gbits/sec
[  1] 4.0000-5.0000 sec  2.13 GBytes  18.3 Gbits/sec
[  1] 5.0000-6.0000 sec  2.23 GBytes  19.2 Gbits/sec
[  1] 6.0000-7.0000 sec  2.32 GBytes  19.9 Gbits/sec
[  1] 7.0000-8.0000 sec  2.27 GBytes  19.5 Gbits/sec
[  1] 8.0000-9.0000 sec  2.23 GBytes  19.1 Gbits/sec
[  1] 9.0000-10.0000 sec  2.24 GBytes  19.2 Gbits/sec
[  1] 0.0000-10.0003 sec  21.3 GBytes  18.3 Gbits/sec
```

```
root@kenisha-virtual-machine:/home/kenisha/4.2D# iperf -c 10.0.0.1 -p 5566 -t 10 > tcp_result.txt
```

Figure 4: Sent packets to h1 from h2

- I then configured the UDP connection. I sent UDP packets from h2 to h1 and the h1 was set to receive them. The file udp_result.txt holds the report summary section.

```
root@kenisha-virtual-machine:/home/kenisha/4.2D# iperf -s -u -p 5566 -i 1
------------------------------------------------------------
Server listening on UDP port 5566
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

Figure 5: UDP

```
  1] local 10.0.0.1 port 5566 connected with 10.0.0.2 port 43031
 ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
  1] 0.0000-1.0000 sec   131 KBytes  1.07 Mbits/sec   0.467 ms 0/91 (0%)
  1] 1.0000-2.0000 sec   128 KBytes  1.05 Mbits/sec   0.442 ms 0/89 (0%)
  1] 2.0000-3.0000 sec   128 KBytes  1.05 Mbits/sec   0.537 ms 0/89 (0%)
  1] 3.0000-4.0000 sec   128 KBytes  1.05 Mbits/sec   0.598 ms 0/89 (0%)
  1] 4.0000-5.0000 sec   128 KBytes  1.05 Mbits/sec   0.385 ms 0/89 (0%)
  1] 5.0000-6.0000 sec   129 KBytes  1.06 Mbits/sec   0.056 ms 0/90 (0%)
  1] 6.0000-7.0000 sec   128 KBytes  1.05 Mbits/sec   0.025 ms 0/89 (0%)
  1] 7.0000-8.0000 sec   128 KBytes  1.05 Mbits/sec   0.024 ms 0/89 (0%)
  1] 8.0000-9.0000 sec   128 KBytes  1.05 Mbits/sec   0.022 ms 0/89 (0%)
  1] 9.0000-10.0000 sec   128 KBytes  1.05 Mbits/sec   0.019 ms 0/89 (0%)
  1] 0.0000-10.0136 sec  1.25 MBytes  1.05 Mbits/sec   0.017 ms 0/895 (0%)
```

```
root@kenisha-virtual-machine:/home/kenisha/4.2D# iperf -c 10.0.0.1 -p 5566 -u -t 10 > udp_result.txt
```

Figure 6: Sent packets to h2 from h1

Figure 7: Output of tcp and udp

- However, once I collected the performance data, I could use Gnuplot to display the results of the performance outcome. I installed Gnuplot on my Ubuntu computer and thereafter developed TCP and UDP performance graphs.

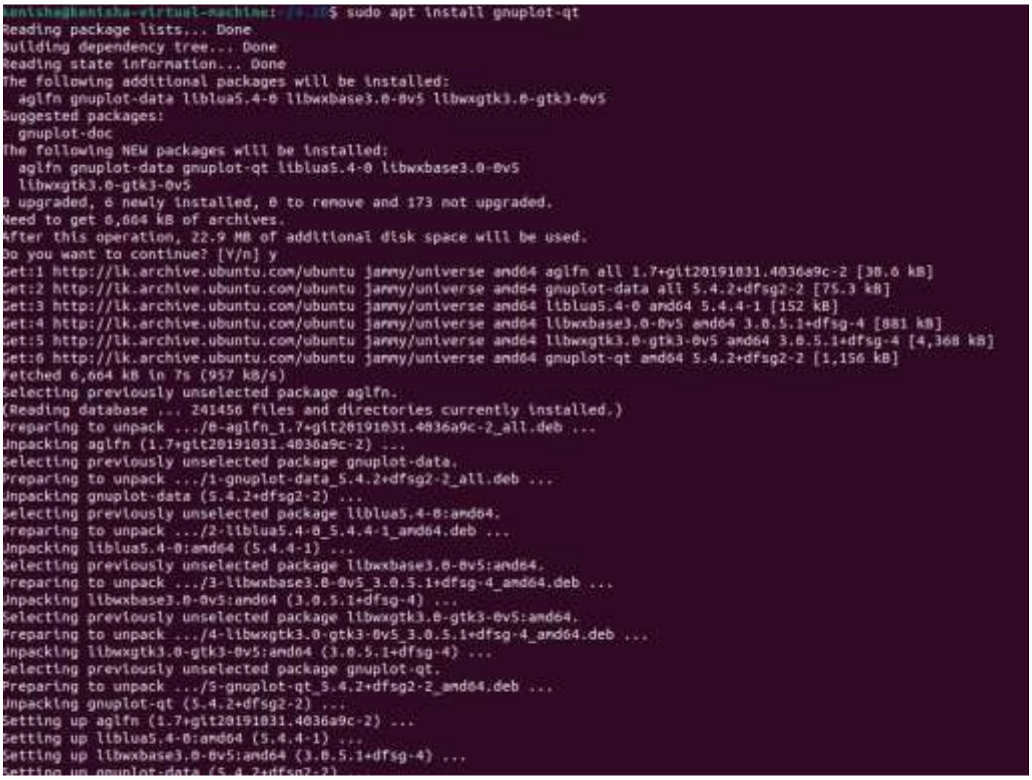


Figure 8: Installation of gunplot

- To ensure all the data were organized properly, I then created two more files which include tcp.dat and udp.dat.

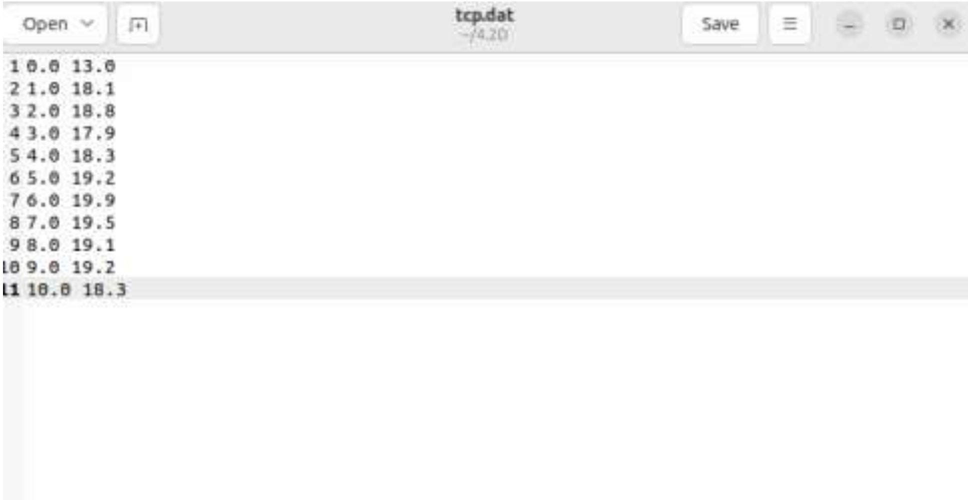


Figure 9: tcp.dat

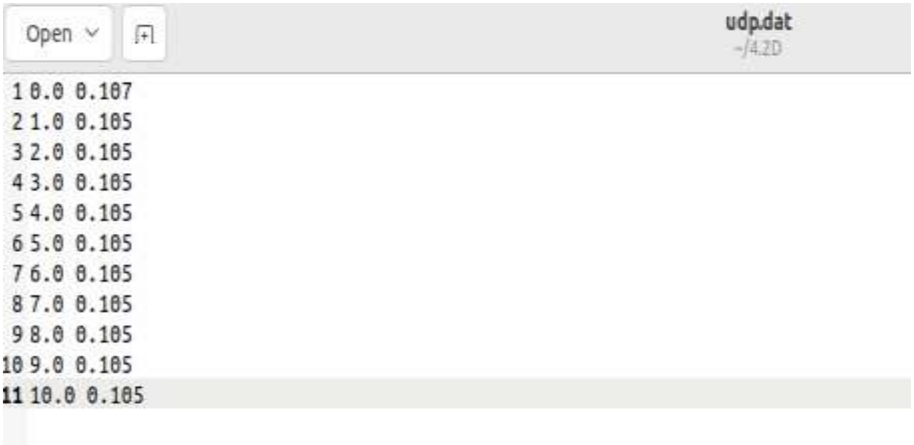


Figure 10: udp.dat

- I then started Gnuplot and plotted each stream type for TCP and UDP individually.

```
gnuplot> set terminal png

Terminal type is now 'png'
Options are 'nocrop enhanced size 640,480 font "arial,12.0" '
gnuplot> set output 'tcp_throughput.png'
gnuplot> set tittle "TCP Throughput"
                 ^
         unrecognized option - see 'help set'.

gnuplot> set title "TCP Throughput"
gnuplot> set xlabel "Time (s)"
gnuplot> set ylabel "Throughput (Gbits)"
gnuplot> plot 'tcp.dat' using 1:2 with lines title "TCP"
```

Figure 11: Draw a plot for tcp data

```
gnuplot> set terminal png

Terminal type is now 'png'
Options are 'nocrop enhanced size 640,480 font "arial,12.0" '
gnuplot> set output 'udp_throughput.png'
gnuplot> set title "UDP Throughput"
gnuplot> set xlabel "Time (s)"
gnuplot> set ylabel "Throughput (Mbits)"
gnuplot> set xrange [0:10.0136]
gnuplot> set yrange [0:*]
gnuplot> plot 'udp.dat' using 1:2 with linespoints title "UDP"
```

Figure 12: Draw a plot for udp data

- After that, shown was the plots of each connection

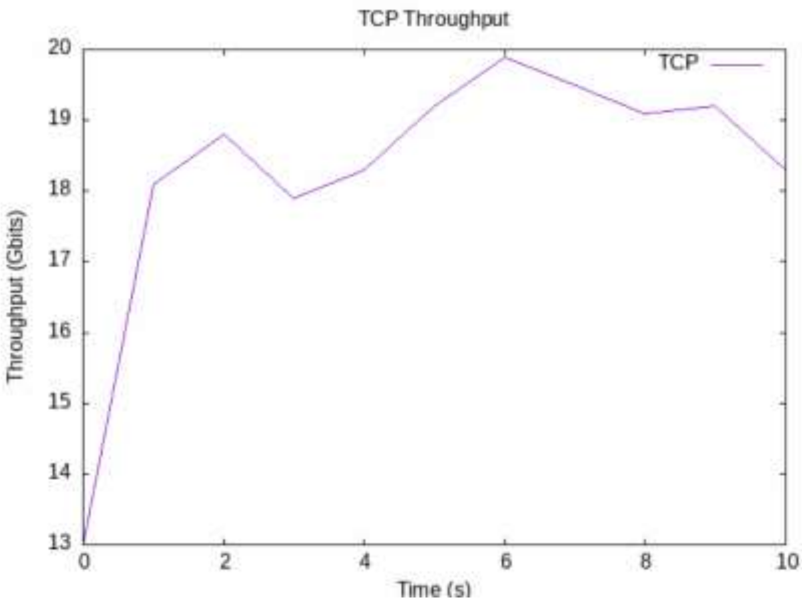


Figure 13: Throughput of TCP

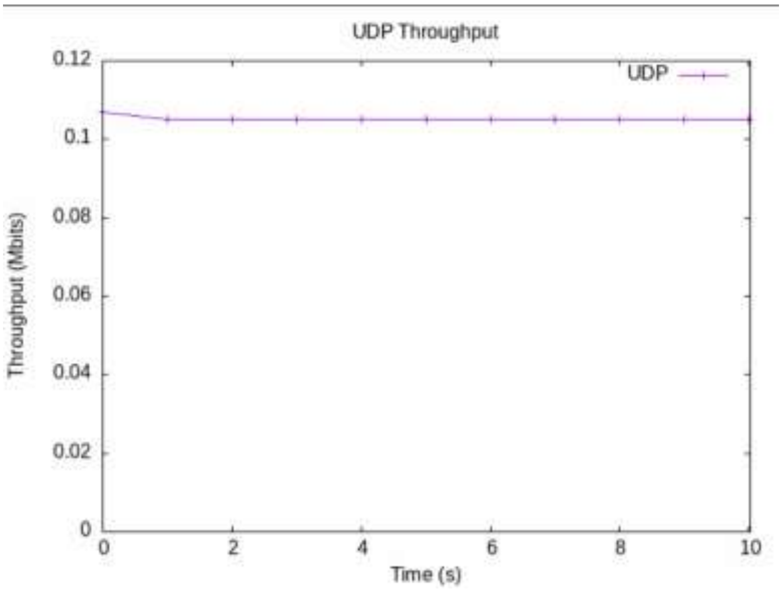
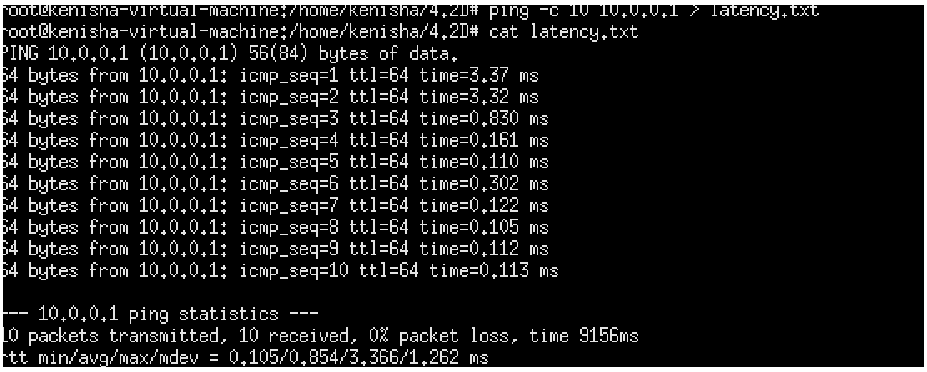


Figure 14: Throughput of UDP

Average Latency

- The Network test resulted in about 1.167 ms of the average latency. RTT or latency is the total amount of time for a packet to travel from its source to its destination and back. The values of the results were obtained as 0.117 ms minimum and 6.803 ms maximum. This average was computed using ten pings.



```
oot@kenisha-virtual-machine:/home/kenisha/4.2D# ping -c 10 10.0.0.1 > latency.txt
oot@kenisha-virtual-machine:/home/kenisha/4.2D# cat latency.txt
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
4 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=3.37 ms
4 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=3.32 ms
4 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.830 ms
4 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.161 ms
4 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.110 ms
4 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.302 ms
4 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.122 ms
4 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.105 ms
4 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.112 ms
4 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.113 ms

-- 10.0.0.1 ping statistics ---
0 packets transmitted, 10 received, 0% packet loss, time 9156ms
tt min/avg/max/mdev = 0.105/0.854/3.366/1.262 ms
```

Figure 15: Testing of ping

- This brings me to understand how sensitive and fast the network is with the low average latency especially when it comes to apps that demand a real-time response like game, VoIPs. Nonetheless, these numbers change constantly so that the difference between the minimum and the largest number is two times more than the mdev, according to the note. In fact, the network, which has an average response time of 034 ms, seems to be running more appropriately. However, I think that there can be some time when the network is populated greatly by the users. These might be occasioned by operational restrictions or might stem from traffic processing in the intermediary devices.

- To this end, the criterion for high performance delivery of 0% inclusive of packet loss was achieved since there was an absence of packet loss to begin with. However, there is a relatively small range to it, and this is where the system is more consistent in returns as far as latency, which is quite critical in some uses, is concerned. That is, its mean latency is one when expressed otherwise. My connective aptitude is high and I have little lag; I am rated 167 ms putting me in the high network connectivity. This is particularly useful in contexts where the response time and data traffic volume are so important.

- I discovered it was best to capture both TCP and UDP results into a single file for Gnuplot visualization. By doing this, I got the throughput of both protocols in one graph making it easier for analysis.

```
gnuplot> set terminal png

Terminal type is now 'png'
Options are 'nocrop enhanced size 640,480 font "arial,12.0" '
gnuplot> set output 'combined_throughput.png'
gnuplot> set title "TCP and UDP Throughput"
gnuplot> set xlabel "Time (s)"
gnuplot> set ylabel "Throughput (Mbits or Gbits)"
gnuplot> set xrange [0:10]
gnuplot> set yrange [0:20]
gnuplot> plot 'tcp.dat' using 1:2 with linespoints title "TCP", \
> 'udp.dat' using 1:2 with linespoints title "UDP"
gnuplot> set terminal png
```

Figure 16: Draw plot for tcp and udp

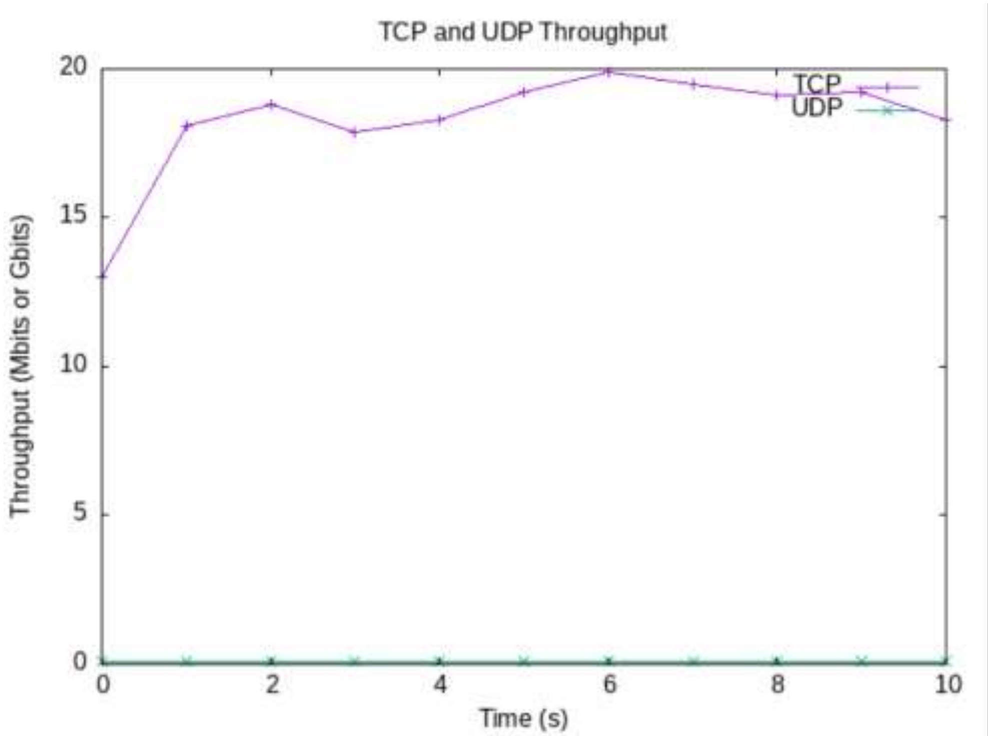


Figure 17: Combination of tcp and udp

Part B

- As for the last figure to represent a job, I choose to use the simple tree topology I used for the third job above in this section. The results presented in the tutorial and the second control point 2C for the network performance measurement. In other words, a specific layer of the network always has a branch factor of 2, which comprises hierarchical tiers of core. And aggregation and edge switch, where the hosts connect to the edge switch.
- In this performance evaluation of this network, I opted to restrict my testing to the ability of the network to support TCP connection to analyze its packet delivery and through put performance. For this exercise, iperf was employed to generate TCP streams to the other hosts and I was then able to determine the bandwidth of the network. Further, I have obtained the chosen hosts' RTT using the ICMP pinging technique. To achieve the aim of automating every activity in part A, I first wrote the improved code from 3.2c.

Open | *tcp_latencytest.py | Save

```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info
import os

class CustomTreeTopo(Topo):

    def __init__(self, n):

        Topo.__init__(self)

        # Create core switch (1 switch)

        core_switch = self.addSwitch('c1')

        # Create aggregation switches (n aggregation switches connected to core)

        agg_switches = []

        for i in range(1, n+1):

            agg_switch = self.addSwitch(f's2{i}')

            self.addLink(core_switch, agg_switch)

            agg_switches.append(agg_switch)

        # Create edge switches (n edge switches per aggregation switch)

        edge_switches = []

        for i, agg_switch in enumerate(agg_switches, start=1):

            for j in range(1, n+1):

                edge_switch = self.addSwitch(f's3{i}{j}')

                self.addLink(agg_switch, edge_switch)

                edge_switches.append(edge_switch)

        # Create hosts (n hosts per edge switch)

        for i, edge_switch in enumerate(edge_switches, start=1):
```

Figure 18: expanded of 3.2c code

- Then, I wrote the following code to achieve the TCP connection and then do the ping test to check the latency.

```
kenisha@kenisha-virtual-machine:~/...$ sudo python3 tcp_latencytest.py
[sudo] password for kenisha:
Enter the value of n (n represents the branching factor): 2
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller ovs-testcontroller udpbwtest mnexec ivs ryu-manager  2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1 -- --if-exists del-br s2
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
( ip link del s1-eth1;ip link del s1-eth2;ip link del s2-eth1;ip link del s1-eth3;ip link del s1-eth3;ip link del s2-eth1;ip link del s2-eth2;ip link del s2-eth3 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
*** Creating network
*** Adding controller
*** Adding hosts:
h11 h12 h21 h22 h31 h32 h41 h42
*** Adding switches:
c1 s21 s22 s311 s312 s321 s322
*** Adding links:
(c1, s21) (c1, s22) (s21, s311) (s21, s312) (s22, s321) (s22, s322) (s311, h11) (s311, h12) (s312, h21) (s312, h22) (s321, h31) (s321, h32) (s322, h41) (s322, h42)
*** Configuring hosts
h11 h12 h21 h22 h31 h32 h41 h42
*** Starting controller
c0
*** Starting 7 switches
c1 s21 s22 s311 s312 s321 s322 ...
Starting iperf server on h2...
Running TCP test...

TCP test output:
```

```
------------------------------------------------------------
Client connecting to 10.0.0.3, TCP port 5566
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 41234 connected with 10.0.0.3 port 5566
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec   296 MBytes  2.49 Gbits/sec
[  1] 1.0000-2.0000 sec   518 MBytes  4.35 Gbits/sec
[  1] 2.0000-3.0000 sec   628 MBytes  5.26 Gbits/sec
[  1] 3.0000-4.0000 sec   770 MBytes  6.46 Gbits/sec
[  1] 4.0000-5.0000 sec   822 MBytes  6.90 Gbits/sec
[  1] 5.0000-6.0000 sec   754 MBytes  6.33 Gbits/sec
[  1] 6.0000-7.0000 sec   798 MBytes  6.69 Gbits/sec
[  1] 7.0000-8.0000 sec   712 MBytes  5.98 Gbits/sec
[  1] 8.0000-9.0000 sec   624 MBytes  5.24 Gbits/sec
[  1] 9.0000-10.0000 sec   685 MBytes  5.74 Gbits/sec
[  1] 10.0000-11.0000 sec   591 MBytes  4.96 Gbits/sec
[  1] 11.0000-12.0000 sec  1.07 GBytes  9.21 Gbits/sec
[  1] 12.0000-13.0000 sec  1.14 GBytes  9.80 Gbits/sec
[  1] 13.0000-14.0000 sec  1.27 GBytes  10.9 Gbits/sec
[  1] 14.0000-15.0000 sec  1.12 GBytes  9.63 Gbits/sec
[  1] 0.0000-15.0163 sec  11.6 GBytes  6.66 Gbits/sec

Measuring latency between h11 and h21...

Latency test output:

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=11.4 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=3.41 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.277 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.228 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.116 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4051ms
rtt min/avg/max/mdev = 0.116/3.078/11.356/4.321 ms

*** Stopping 1 controllers
c0
*** Stopping 14 links
..............
*** Stopping 7 switches
c1 s21 s22 s311 s312 s321 s322
*** Stopping 8 hosts
h11 h12 h21 h22 h31 h32 h41 h42
*** Done
```

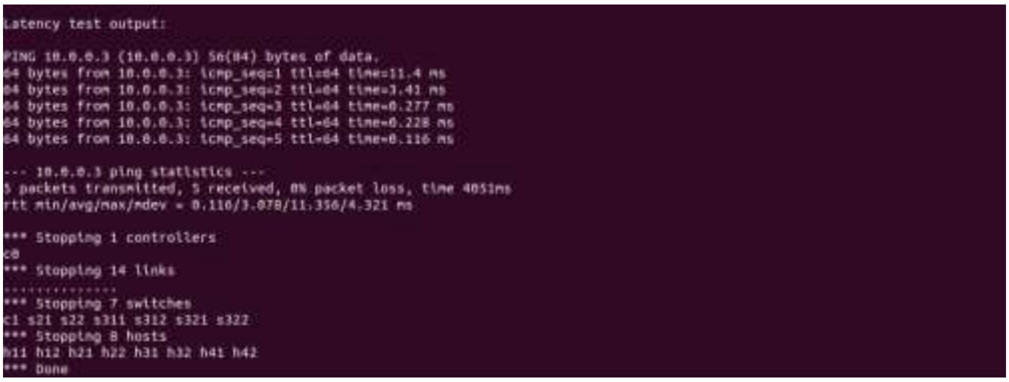
Figure 19: Running the code

- Then, the data into a separate file and the Gnuplot script was generated to produce the graph as given below.

tcp_latency.txt

```
0.0 2.49
1.0 4.35
2.0 5.26
3.0 6.46
4.0 6.90
5.0 6.33
6.0 6.69
7.0 5.98
8.0 5.24
9.0 5.74
10.0 4.96
11.0 9.21
12.0 9.80
13.0 10.9
14.0 9.63
15.0 6.66
```

Figure 20: Output of ping test



```
Latency test output:

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=11.4 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=3.41 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.277 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.228 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.116 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4051ms
rtt min/avg/max/mdev = 0.116/3.078/11.356/4.321 ms

*** Stopping 1 controllers
c0
*** Stopping 14 links
..............
*** Stopping 7 switches
c1 s21 s22 s311 s312 s321 s322
*** Stopping 8 hosts
h11 h12 h21 h22 h31 h32 h41 h42
*** Done
```

Figure 21: Ping Test

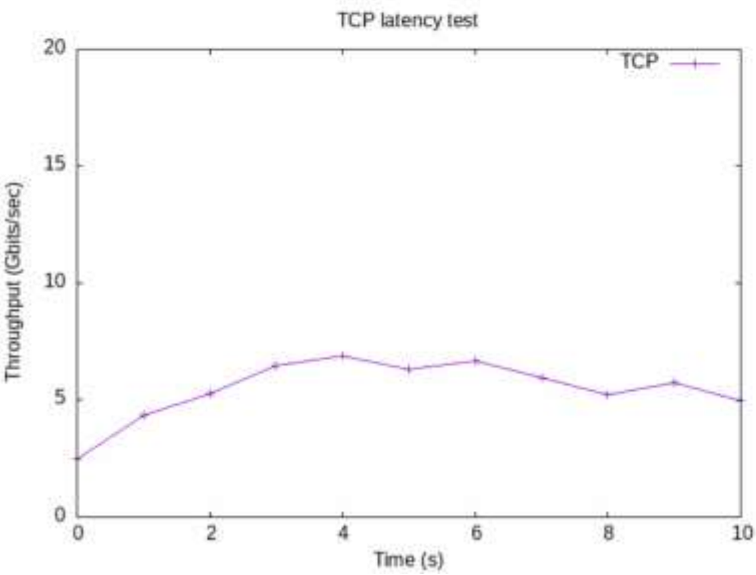


Figure 22: plot of TCP

- The latency was then established to be approximately around 2. 998 ms using the results of the ping test between H11 and H21. Therefore the latency value for the network latency in milliseconds varies from as low as 084 ms to a high of 11 ms. the values were 355 ms of mean reaction time with minimal SD that ranged 4 ms for everyone.

- Still, the average of all latency is less than the average which refers that in overall the network is quite fast which is beneficial in applications that are largely based on the real-time communications only. But sometimes it is measured that leads to a few higher latency values particularly in 11. As much as it is important to appreciate that this experiment incorporates a measurement of 355 ms, there is basis to doubt the implication of this time on the availability of the network. Among them can be mentioned such ones that may occur as a result of a network congestion or delay during message transfer between two nodes.

- The mean latency of the network is, as a whole, low, however it also seems to be highly volatile. This underlines that the network characteristics should be checked regularly even more so if an application might require time synchronization.

Part C

Q1. Give 5 valid reasons that why network performance evaluation is important in non-attack scenarios. Explain your answer (each point) with 2 examples.

1. **Capacity Planning and Scalability**

- Performance analysis of the network is critical if one is to determine the patterns of consumption on the network today and how the space for growth can be created. In order to accommodate new needs, this assists in increasing the capability of the network infrastructure.

  - **Example**: **Cloud Services:** These are networks like AWS and Azure, they are able to get network performance data to see how they can do better by serving more potential clients. That way they can develop their infrastructure to meet the demand that is created by such patterns suitably. (AWS, 2023)

  - **Corporate Networks:** The Company may need a new network if the hiring system of the company has been improving. Taken from performance reports on current bandwidth usage, performance evaluations assist IT departments in ensuring a future increase in bandwidth and the necessary network equipment.

2. **A Service Level Agreement or SLA are contractual stipulations that define certain service levels that need to be provided.**

- In light of the fact that the SLAs are expected to define the levels of service that the customers, in particularly, desire, it is imperative to evaluate the network performance so as to determine the feasibility of these levels. In this is included monitoring of the networks, response time and up time of the networks.

    - **Example: Contracts between businesses (B2B):** In B2B contracts of business organizations, the performance standards to be attained are usually set. Reviewing performance from time to time makes it possible to achieve those goals that in turn improves business relationships. (Kempter, 2023)

    - **Managed Service Providers (MSPs):** This applies because, MSPs are compelled by their clients' SLAs to monitor their networks' health in an attempt to meet necessary and desirable network uptime and response time. This assists to retain the confidence of the client and prevent penalties.

3. Enhancing the Use of Resources

- As mentioned earlier, benefits in performance help in the discovery of ways in enhancing the use of the company network to increase possibly the use of links and bandwidth.

  - **Example: Bandwidth Allocation:** Given herein are some of the possible results that may be considered when assessing the overall performance of business networks; such include: This assists the network administrators in deciding on which traffic kind should be given priority over the others or which traffic type should be given the additional unused bandwidth in order to optimize the usage of the business network.

  - **Load balancing:** It assists servers to identify their level of utilization in a data center network, and consequently data center network. As the function allows data centers to balance loads about traffic pattern to make sure that some servers will not get overloaded and at the same time, making the best out of the system. (Flare, 2024)

4. **Making Certain the Best User Experience**

- In the opinion of this author, the uptake of network performance assessments can afford network service users ease. The kind of problems that may hinder the quality of the user interactions could be identified by accounting for the identified features of the observed network.

    - **Example: Online gaming:** They include – performance testing Preventing lag and slow movement is critical in most games and other graphical uses of the computer. In other words, for the kinds of applications that FPVD delivers, those that are synchronous and involve interaction with others, such as games like Call of Duty or Fortnite, low latency is required. The performance evaluations needed throughout the network mostly help in keeping the responsiveness and speed needed on the favorable side.

    - **Streaming Services:** Performance analysis on a network ensures there is no break in the video flow across distribution channels which include YouTube and Netflix. As pointed out above, it is possible to increase content delivery effectiveness and, correspondingly, the satisfaction of the provider by quantifying such basic parameters as bandwidth, latency, and packet loss.

5. Finding and Fixing Bottlenecks

- To assist with the need for these adjustments, network performance evaluation aids in determining occasions that the network may be struggling or seizing to provide optimum performance.

  - **Example: Application Performance:** If a web application is slow, one can, therefore, use the performance evaluation to conclude that the network has a high latency or loss packets. Thus, it becomes possible to improve the application's speed and convenience for the users if all these problems are addressed.

  - **Database Access:** In centrally positioned databases, performance evaluation procedures can raise the question of slow data accessibility. Solving these problems makes it easier to look for solutions for expanding such networks or even buying new hardware to make databases faster. (Sheldon, 2021)

Conclusion

- In this research, operation of an actual networks is employed along with the Iperf and the Gnuplot to establish the utility of evaluating network performance particularly under the non-attack scenario. Through analyzing TCP and UDP, one is able to identify through output concerning throughput, latency and overall efficiency of the network. These assessments are very much important to determine the capacity, services level agreements, resource utilization and optimization, measurements for enhanced users' satisfaction and identification of bottlenecks. These findings demonstrate the applicability of network performance analysis in practical scenarios and stress on prerequisite continual monitoring, which in effect provides stable and efficient network substrates.

Reference

- Netflix Research

https://research.netflix.com/

- Cloud Computing Infrastructure Explained – AWS

https://aws.amazon.com/what-is/cloud-infrastructure/

- Sheldon, R. (2022) SQL Server performance tuning

https://www.red-gate.com/simple-talk/databases/sql-server/database-administration-sql-server/sql-server-performance-tuning-nine-best-practices/

- Kempter, S.Service Level Management

Service Level Management | IT Process Wiki

- Global Server load balancing

https://www.cloudflare.com/en-gb/learning/cdn/glossary/global-server-load-balancing-gslb/