APRIL 17, 2023

PORTFOLIO 1

DATA2410

SURLAND, GLENN ANDRE HANSEN
S354604
Oslo Metropolitan University

Table of contents

1 Introduction

Network performance measurement and analysis is crucial for optimizing network resources and ensuring efficient and reliable data transmission. *Simpleperf* is a simple network measurement tool designed to send and receive packets between a client and server using sockets. The objective of this project is to evaluate the performance of a virtual network topology in Mininet using Simpleperf and to evaluate the performance of Simpleperf under different conditions and configurations. This project aims to measure network performance, analyze the impact of network parameters on the network's throughput and identify potential issues.

Simpleperf and other network monitoring and analysis tools will be used to measure and analyze the performance of the network created by the script portfolio_topology.py. This project provides value by providing a simple and flexible tool for measuring network performance in a virtual network using Mininet. Some relevant work in this field includes articles describing the network performance measurement tool IPerf, as well as other network monitoring and analysis tools.

The article "Performance Monitoring of Various Network Traffic Generators" by Samad S. Kolahi, Shaneel Narayan, Du. D. T. Nguyen, and Yonathan Sunarto compares the performance of four network traffic generators (IPerf, Netperf, D-ITG, and IP Traffic) in a laboratory environment using two computers with Windows operating systems connected via a 100 Mbps link. The study measures TCP traffic on the link for various payload sizes and analyzes the results using the different monitoring tools. The authors found that the tools can produce significantly different results, with IPerf measuring the highest bandwidth and IP Traffic measuring the lowest for a packet size. However, the results varied for different packet sizes, highlighting the importance of careful tool selection and ongoing performance monitoring. The article provides valuable insights into the capabilities of different traffic generators and emphasizes the need for accurate and reliable performance monitoring in network testing and troubleshooting.

The article "On the Unreliability of Network Simulation Results from Mininet and iPerf" by Benjamin Hardin, Douglas Comer, and Adib Rastegarnia examines the accuracy and reliability of two commonly used tools in network simulation, Mininet and iPerf. The authors argue that these tools may not accurately reflect the behavior of real-world networks and may lead to incorrect conclusions. The article highlights surprising anomalies in the results produced by Mininet and iPerf and makes recommendations for configuring Mininet to avoid some of the anomalies. The authors conclude that the research findings have significant implications for the networking research community and industry, and emphasize the need for caution and careful validation when using Mininet and iPerf to assess network architectures.

As concluded in the articles, there are several limitations to the approach, including the fact that the experiments are limited to a virtual network environment and may not accurately reflect real-world network conditions. Additionally, there may be limitations in terms of the scalability of our approach and the number of network parameters that can be evaluated. However, the outcome will include a better understanding of the impact of network parameters on network performance, as well as the development of a simple and flexible tool for measuring network throughput that can be used in a wide range of network environments.

This report evaluates the performance of a network using the Simpleperf tool. The report begins with an introduction to Simpleperf and a discussion of its code structure. The experimental setup, which includes the Mininet script topology, is then described. Various network tools and performance metrics are used to evaluate the network's performance in five different test cases. These test cases include measuring bandwidth with IPerf in UDP mode, evaluating link and path latency and throughput, studying the effects of multiplexing and latency, and investigating the effects of parallel connections. The report concludes with a summary of the results and a discussion of their implications and limitations.

2 Simpleperf

2.1 System process flow

The Simpleperf tool is designed to measure network throughput by sending and receiving packets between a client and a server using sockets. The tool is implemented in Python, and the building blocks of Simpleperf include a client mode and a server mode. In server mode, the server listens for incoming connections from clients and handles the connections in separate threads. In client mode, the client connects to the server and sends packets of a specified size at a specified rate.

The communication between the server and client is implemented using the socket module in Python. The server listens for incoming connections on a specified port and accepts incoming connections using the accept method. Each client connection is handled in a separate thread, and the thread is responsible for receiving packets from the client, tracking the number of bytes received, and printing statistics at specified intervals.

The client connects to the server using the connect method and sends packets of a specified size at a specified rate using the sendall method. The client also tracks the number of bytes sent and prints statistics at specified intervals. Once the client has finished sending packets, it sends a "BYE" command to the server to indicate that the connection should be closed. The server responds with an "ACK" message to confirm that the server received the ACK message.

Overall, Simpleperf is a flexible and simple tool for measuring network throughput that can be used in a variety of network environments. The communication between the client and server is implemented using standard socket communication, and the tool provides a number of configuration options to customize the measurement process. The following paragraphs will go more in depth of each function/building block that the system consists of in chronological order.

2.2 Code structure

print_statistics()

The *print_statistics()* function in the server is responsible for printing the statistics of the network performance. The function takes four input parameters: *client_address, bytes_sent, interval_number, interval_size, and format.*

The function first checks the format argument and converts the bytes to the appropriate units based on the user's preference. If format is "B", then the bytes are left as they are. If format is "KB", then the bytes are converted to kilobytes. If format is "MB", then the bytes are converted to megabytes.

The function then constructs the id string by concatenating the IP address and port number of the client socket. It also constructs the interval string by concatenating the start and end times of the interval. Finally, it calculates the bandwidth using the formula: *bytes_sent / (CHUNK_SIZE * CHUNK_SIZE * interval_size) * 8* and formats the result to one decimal place.

The function then prints the statistics in a well-formatted string with appropriate spacing and alignment using the print() function.

server()

The server function is responsible for setting up a socket that listens for incoming connections from clients. The function takes in the arguments specified by the user when running the tool and uses them to configure the socket connection.

First of all, the function creates a socket using the socket library in Python and binds it to the specified IP address and port number using the bind() method. It then sets the socket to listen for incoming connections using the listen() method.

Once the socket is listening, the server enters a loop where it waits for incoming client connections using the accept() method. When a client connects, the server creates a new thread to handle the client and starts the thread using the Thread() method. The thread calls the handle_client() function to handle the client connection.

The server() function also prints out information to the console to indicate that the server is running and listening for incoming connections. This information includes the IP address and port number on which the server is listening.

handle_client()

The *handle_client* function is responsible for handling client connections on the server side.

When a client connects to the server, the handle_client function is executed in a separate thread to handle that connection. It receives the client socket as an argument and sets up some initial variables such as *bytes_received, client_address,* and *start_time*.

The function then enters an infinite loop to receive data from the client in chunks of CHUNK_SIZE (1000). It keeps track of the total number of bytes received in the bytes_received variable.

The loop checks for a specific string of bytes, "BYE", at the end of each chunk received from the client. If the "BYE" string is received, the function sends an acknowledgement to the client and calculates the end time of the connection. It then prints the statistics for the connection using the *print_statistics* function.

Finally, the function closes the client socket.

client()

The client function is responsible for running the client mode of the simpleperf tool. It connects to a remote server and sends data to it for a specified period of time or until a specified amount of data has been sent. The function takes in command line arguments through the args parameter.

The function begins by creating a socket and connecting to the remote server using the IP address and port number specified in the command line arguments. It then sends a START command to the server to begin sending data. The data to be sent is a chunk of bytes of size CHUNK_SIZE.

The function enters a loop that sends data to the server until the specified time limit or byte limit is reached. During each iteration of the loop, it sends the data and increments the total bytes sent by the CHUNK_SIZE. It also calculates the time elapsed since the start of the connection.

If an interval has been specified in the command line arguments, the function prints statistics at regular intervals. It keeps track of the bytes sent during the interval and the number of intervals that have passed. When the time for an interval has elapsed, the function prints the statistics for that interval using the print_statistics function.

Once the loop has finished, the function sends a BYE command to the server to signal the end of the data transmission. It then waits for an acknowledgement from the server before closing the socket. Finally, the function prints the total statistics for the data transmission using the print_statistics function.

if __name__ == '__main__'

In Simpleperf, the if __name__ == '__main__' statement serves as the entry point to the program. It begins by creating an ArgumentParser object from the argparse module, which allows the program to parse command-line arguments.

Next, several command-line arguments are defined, including flags for enabling server or client mode, specifying the port number, setting the output format, and setting various parameters for the client mode. Here is a full list of the command-line arguments the user can use.

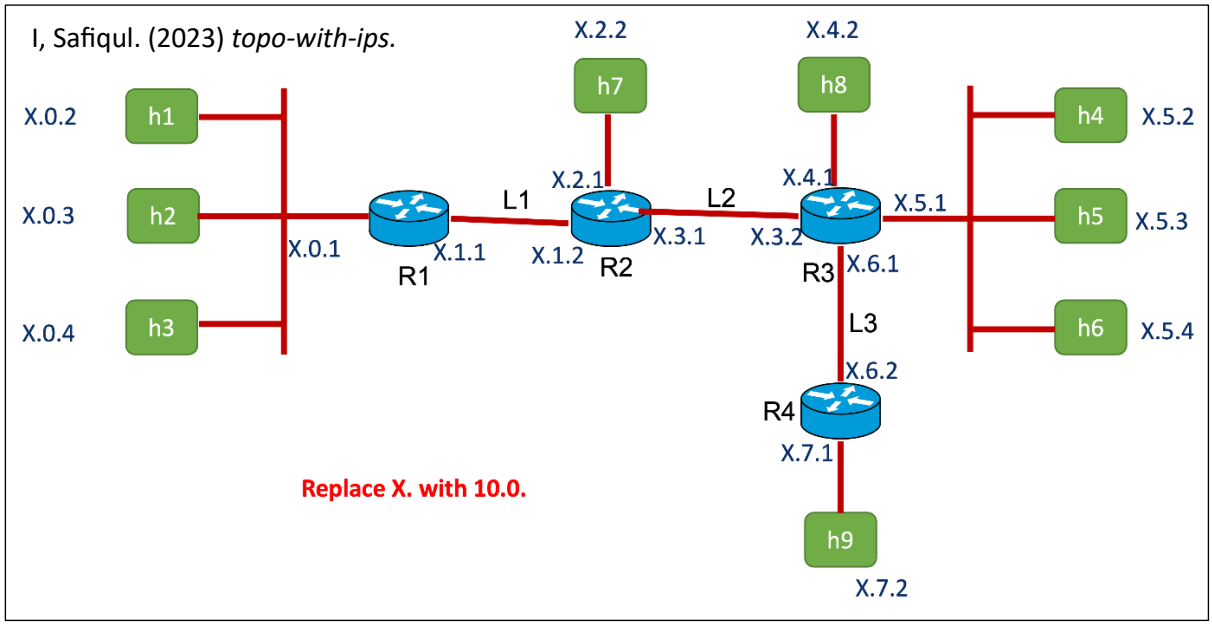
Argument	Flag	Description
--server	-s	Enables server mode. This means that the program will run in server mode and listen for incoming connections from clients.
--client	-c	Enables client mode. This means that the program will run in client mode and connect to a server to send data.
--port	-p	Specifies the port number on which the server should listen or the client should connect.
--format	-f	Specifies the format in which data should be displayed. This flag takes one of three possible values: "B" (bytes), "KB" (kilobytes), or "MB" (megabytes). Default: MB.
--bind	-b	Specifies the IP address of the server interface. This flag is only used in server mode.
--serverip	-I	Specifies the IP address of the server. This flag is only used in client mode.
--time	-t	Specifies the total duration in seconds for which data should be generated.
--interval	-i	Print statistics per interval seconds
--parallel	-P	Specifies the number of parallel connections to create.
--num	-n	Specifies the total number of bytes to transfer, in B (bytes), KB (kilobytes), or MB (megabytes).

After defining the command-line arguments, the program uses the argparse module to parse the command-line arguments supplied by the user. Finally, the program checks whether the user specified server or client mode using the args.server and args.client flags. If the user specified server mode, the program calls the server function with the parsed command-line arguments. If the user specified client mode, the program either creates a single client connection or multiple parallel connections based on the args.parallel flag and calls the client function with the parsed command-line arguments.

3 Experimental setup

3.1 Mininet script topology

The virtual network topology consists of multiple subnets connected through routers, switches, and hosts. In this section, I will describe each subnet that makes up the network.



Subnet A:

- Three hosts: h1 (10.0.0.2/24), h2 (10.0.0.3/24), and h3 (10.0.0.4/24)
- One switch: s1
- One router: r1 (10.0.0.1/24)

Hosts h1, h2, and h3 are connected to switch s1, which in turn connects to router r1.

Subnet B:

- Routers r1 (10.0.1.1/24) and r2 (10.0.1.2/24) are directly connected.

Subnet C:

- Router r2 (10.0.2.1/24) connects to host h7 (10.0.2.2/24).

Subnet D:

- Routers r2 (10.0.3.1/24) and r3 (10.0.3.2/24) are directly connected.

Subnet E:

- Three hosts: h4 (10.0.5.2/24), h5 (10.0.5.3/24), and h6 (10.0.5.4/24)
- One switch: s2
- One router: r3 (10.0.5.1/24)

Hosts h4, h5, and h6 are connected to switch s2, which in turn connects to router r3.

Subnet G:

- Routers r3 (10.0.6.1/24) and r4 (10.0.6.2/24) are directly connected.

Subnet I:

- Router r4 (10.0.7.1/24) connects to host h9 (10.0.7.2/24).

In summary, the virtual network consists of four routers (r1, r2, r3, and r4), two switches (s1 and s2), and nine hosts (h1 through h9). The routers are responsible for forwarding packets between subnets, and the switches are responsible for forwarding packets between hosts within a subnet.

4 Performance evaluations

4.1 Network tools

VirtualBox: VirtualBox is a virtualization platform that allows users to run multiple operating systems simultaneously on a single physical machine. In this experiment, VirtualBox was used to emulate a virtual machine running Ubuntu, which made it possible to run the Mininet network emulator as Mininet is not supported on Windows.

Mininet: Mininet is a network emulator that allows users to create virtual networks for testing and development purposes. Mininet provides a simple Python API for creating and managing network topologies, as well as a command-line interface for interacting with the network. In this experiment, Mininet was used to create and perform tests on the virtual network created by the portfolio-topology.py script.

Xterm: Xterm is a terminal emulator that allows users to run terminal-based applications in graphical mode. In this experiment, xterm was used in Mininet to open terminal windows for each individual host and router in the network.

IPerf: IPerf is a tool for measuring network throughput and performance. The program can be used to generate traffic between hosts in the network and measure the bandwidth, latency, and other performance metrics. In this experiment, IPerf was used to measure bandwidth in test case 1 by running it on three separate client-server pairs in UDP mode.

Ping: Ping is a tool for testing network connectivity by sending ICMP echo request packets to a remote host and measuring the response time. Ping can be used to test the connectivity between hosts in the network and diagnose network issues. In this experiment, Ping was mainly used to measure the RTT between hosts and routers in several test cases.

4.2 Performance metrics

Bandwidth: Bandwidth is a measure of the amount of data that can be transferred over a network connection in a given period of time. The higher the bandwidth, the faster data can be transferred. I measured the bandwidth of data transfer between the client and server using IPerf and simpleperf to evaluate the performance of the network.

Latency: Latency is the time it takes for a packet to travel from the sender to the receiver and back again. Low latency is important for real-time applications such as video conferencing or online gaming. In this assignment, I measured latency using the Ping tool to observe RTT between hosts and routers.

Jitter: Jitter is the variation in latency over time. High jitter can cause problems for real-time applications, as it can result in inconsistent performance. Jitter was measured in test case 1 between three different host pairs using IPerf.

Packet loss: Packet loss occurs when packets are dropped during transmission. This can result in degraded performance or incomplete data transfer. During test case 1, packet loss was observed to find the highest stable Mbps without any loss.

In summary, bandwidth, latency, jitter, and packet loss are all important network performance metrics that are commonly used to evaluate the quality and reliability of network connections. During the experiments, all of the mentioned network performance metrics are evaluated and reported in the following segments of the assignment.

4.3 Test case 1: Measuring bandwidth with IPerf in UDP mode

4.3.1 Introduction

IPerf is a command-line tool used for measuring network bandwidth and performance. It can be used to generate TCP and UDP traffic, and in this case, It will be used in UDP mode as specified. UDP stands for User Datagram Protocol, which is a connectionless protocol that doesn't guarantee packet delivery or order.

Measuring bandwidth with IPerf in UDP mode involves running the IPerf tool on two machines - a server and a client. The server listens for incoming traffic, and the client sends traffic to the server at a specified rate. The tool then measures the amount of traffic sent and received and the throughput. In this experiment, I will be running three separate IPerf tests using IPerf in UDP mode and a specified bandwidth of 'X'M between different client-server pairs using xterm. Specifically, I will be testing the bandwidth between the host pairs h1-h4, h1-h9, and h7-h9.

4.3.2 Results and discussion

Test	Interval	Transfer	Bandwidth	Jitter	Lost/Total
H1-H4	0.0000-10.0008 sec	31.3 MBytes	26.2 Mbits/sec	0.128 ms	0/22294 (0%)
H1-H9	0.0000-9.9992 sec	12.5 MBytes	10.5 Mbits/sec	0.174 ms	0/8920 (0%)
H7-H9	0.0000-10.0008 sec	12.5 MBytes	10.5 Mbits/sec	0.329 ms	0/8920 (0%)

For the H1-H4 test, the transfer value was 31.3 MBytes, which indicates the total amount of data transferred during the test, and the bandwidth value was 26.2 Mbits/sec, which indicates the average rate at which data was transferred during the test. This test had the highest bandwidth value among the three tests conducted.

For the H1-H9 and H7-H9 tests, the transfer values were 12.5 MBytes, and the bandwidth values were 10.5 Mbits/sec. These values were lower than the H1-H4 test and indicate that the network was not able to transmit data at a high rate between h1 and h9 and between h7 and h9.

The jitter values for all three tests are very low, indicating that the network is providing a stable and consistent network performance. Jitter is the variation in the delay between packets arriving at the receiver, and it can cause disruptions in real-time applications such as gaming and video and audio streaming. A low jitter value is desirable for such applications as it ensures a smooth and uninterrupted experience for the users.

All three tests show no packet loss, indicating that the transfer between the specific pairs is stable and reliable on the specified rate of 26.2 Mbits/sec (H1-H4) and 10.5 Mbits/sec (H1-H9 & H7-H9). I performed several tests on different bandwidth rates ('X'M) to figure out what the highest rate of Mbps were without any datagram/packet loss.

To conclude, we can observe that the bandwidth between H1-H4 is the highest among the three tests, which could be due to the fact that H1-H9 and H7-H9 goes through the L3 link and connects to the R4 router. L3 and/or R4 could potentially introduce some delay and limit the bandwidth capacity.

4.3.3 Questions

1. Which rate (X) would you choose to measure the bandwidth here? Explain your answers.

To choose a rate (X) to measure the bandwidth, it is necessary to consider the maximum available bandwidth of the network, as well as any rate limiting factors that may be in place. The specific rate to choose would depend on the characteristics of the network, such as the number of routers, switches, and hosts, as well as the link capacities and latency.

In this particular case I would limit H1-H4 at around 25 Mbps and H1-H9 & H7-H9 at around 10 Mbps. I personally used the trial and error approach to find the highest stable mbps rate without any packet/datagram loss.

2. If you are asked to use iPerf in UDP mode to measure the bandwidth where you do not know anything about the network topology, what approach would you take to estimate the bandwidth? Do you think that this approach would be practical? Explain your answers.

When using iPerf in UDP mode to measure the bandwidth without prior knowledge of the network topology, one approach would be to run a series of tests with different payload sizes and vary the UDP throughput until the network starts to drop packets. The highest throughput value before packet loss occurs would give an estimate of the maximum achievable bandwidth in the network. As specified earlier, this is the approach I used to find the right 'X'M that had no packet loss, however this approach may not be practical for very large networks or networks with highly variable traffic patterns, as it would require significant time and resources to perform a sufficient number of tests to accurately estimate the available bandwidth.

Another limitation is that it requires access to both ends of the network and the ability to run iPerf on those endpoints. In some network environments, such as those with restricted access or where endpoints are not under the control of the experimenter, this may not be possible. Additionally, this approach assumes that the network is stable and not congested during the measurement. If the network is highly variable or prone to congestion, this approach may not provide an accurate estimate of the maximum achievable bandwidth.

4.4 Test case 2: Link latency and throughput

4.4.1 Introduction

Test case 2 involves measuring the link latency and throughput for each of the three individual links between routers in the network topology. This information will help us identify the performance characteristics of each link and understand how well they are functioning. The latency and throughput will be measured using ping and simpleperf.

4.4.2 Results and discussion

Ping latency

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
L1 (R1-R2)	24043ms	20.110	20.695	25.498	1.009 ms
L2 (R2-R3)	24031ms	40.109	40.525	41.468	0.400 ms
L3 (R3-R4)	24036ms	20.104	20.582	21.586	0.413 ms

The ping latency tests show that the RTT (Round Trip Time) between the routers on each link is quite low, ranging from 20.104 to 40.525 milliseconds. This indicates that the links have low latency and are performing well. The maximum RTT observed was 41.468 ms, which is still relatively low and indicates that there are no significant delays or bottlenecks in the network. The standard deviation (Mdev) of the RTT measurements is also low, ranging from 0.4 to 1.009 ms, indicating that the latency measurements are consistent and reliable. Overall, the ping latency tests show that the links in the network have good performance and are not a bottleneck for traffic.

L1 and L3 have significantly lower RTT with 20.695 ms and 20.582 ms compared to 40.525 ms. These results show that the latency between R2 and R3 is higher compared to the other links, which is expected if we take a look at the Mininet script that has a configured delay of 20 ms for the link between R2 and R3. The other links have a configured delay of 10 ms, resulting in lower latency.

We can also see that the RTT values for L1 are more spread out or have more variability compared to L2 and L3 by looking at the mean deviation. This could be due to various factors such as network congestion, packet loss, and jitter.

Simpleperf throughput

Test	Interval	Recieved	Rate
L1 (R1-R2)	0.0 - 25.1	55.44 MB	17.7 Mbps
L2 (R2-R3)	0.0 - 25.0	51.22 MB	16.4 Mbps
L3 (R3-R4)	0.0 - 25.2	36.13 MB	11.5 Mbps

From the results of the throughput tests, we can see that the bandwidth between R1-R2 is the highest at 17.7 Mbps, followed by R2-R3 at 16.4 Mbps and R3-R4 at 11.5 Mbps. This suggests that the link between R1-R2 has the highest capacity, while the link between R3-R4 has the lowest capacity.

In summary, L1 has the highest bandwidth and relatively low latency, but with more variability in latency compared to L2 and L3. L2 has the second-highest bandwidth and the highest latency, while L3 has the lowest bandwidth and relatively low latency, similar to L1.

4.5 Test case 3: Path Latency and throughput

4.5.1 Introduction

Test case 3 involves measuring the path latency and throughput for three different scenarios involving communication between various hosts in the network. The primary objective is to understand the network performance in terms of latency and throughput when hosts communicate over different paths. To achieve this, I will conduct the following tests:

- Host 1 (H1) communicates with Host 4 (H4)
- Host 7 (H7) communicates with Host 9 (H9)
- Host 1 (H1) communicates with Host 9 (H9)

For each test, I will use the 'ping' tool to measure the path latency, and simpleperf to measure the path throughput. The tests will be run with consistent parameters (25 packets/a duration of 25 seconds) to ensure comparability of results. The average Round Trip Time (RTT) and measured throughput values will be analyzed to determine the network performance under each scenario. This analysis will help to identify potential bottlenecks and factors that may be affecting the communication between different hosts in the network.

Before performing the tests, I expect that the H1-H9 path will have the lowest bandwidth and highest latency due to the fact that it involves the most path complexity going through a switch and 4 routers. If we take a look at the results from the previous test cases, we can also notice that the L3 link is significantly slower, which will have an impact on the H1-H9 and H7-H9 results. Therefore, I predict H1-H4 to have the lowest delay and highest throughput.

4.5.2 Results and discussion

Path latency

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
H1-H4	24035ms	60.350	61.315	63.961	0.856 ms
H1-H9	24033ms	80.397	81.292	82.421	0.547 ms
H7-H9	24026ms	60.245	60.948	62.424	0.514 ms

Simpleperf throughput

Test	Interval	Recieved	Rate
H1-H4	0.0 - 25.2	48.89 MB	15.5 Mbps
H1-H9	0.0 - 25.1	28.11 MB	9.0 Mbps
H7-H9	0.0 - 25.1	44.55 MB	14.2 Mbps

From these results, we can observe that H1-H9 has a higher latency and lower bandwidth compared to H1-H4 and H7-H9 as expected. This is likely due to the increased number of routers (4 routers) that the H1-H9 path traverses compared to H1-H4 (3 routers) and H7-H9 (3 routers). The additional router in the H1-H9 path introduces extra processing and forwarding delays, resulting in higher latency. Additionally, it traverses the L3 link, which is significantly slower than the other links.

The results also presents that the H1-H4 and H7-H9 paths have similar latencies and throughputs, as both paths traverse the same number of routers (3 routers). I expected the H1-H4 path to be faster than the H7-H9 path, but the results are almost identical. The H1-H4 path do not pass through the L3 link with higher delay, but it goes through two additional switches, so the throughput rates remain relatively similar to H7-H9.

4.6 Test case 4: effects of multiplexing and latency

4.6.1 Introduction

In test case 4, the effects of multiplexing and latency will be tested. The experiment will be conducted in several steps using pairs and groups of hosts that will communicate simultaneously. All of the pairs of hosts will be measured using ping and simpleperf. I will also be performing each of the host connection individually, so I have a basis to make performance comparisons. The pairs are:

1. h1-h4 and h2-h5
2. h1-h4, h2-h5, and h3-h6
3. h1-h4 and h7-h9
4. h1-h4 and h8-h9

h1-h4 and h2-h5: Both pairs of hosts are communicating through the same routers (R1, R2, and R3). As a result, there will likely be increased contention for network resources on these routers and the links between them. This could lead to increased latency and reduced bandwidth for both pairs of hosts.

h1-h4, h2-h5, and h3-h6: All three pairs of hosts share the same routers (R1, R2, and R3). With more simultaneous communication, contention for network resources on these routers and links between them is expected to be even higher. This will likely result in increased latency and reduced bandwidth for all three pairs of hosts.

h1-h4 and h7-h9: The h1-h4 communication path involves routers R1, R2, and R3, while the h7-h9 path involves routers R2, R3, and R4. Both pairs share R2 and R3 routers, so there might be some contention for network resources, which could lead to increased latency and reduced bandwidth. However, the impact might be less than in the first two tests since the paths are partially separate.

h1-h4 and h8-h9: The h1-h4 communication path involves routers R1, R2, and R3, while the h8-h9 path involves routers R3, R4. Both pairs share only the R3 router, which might result in some contention for network resources, but the impact is likely to be less pronounced than in the first three tests. The latency and bandwidth might be slightly affected, but the impact should be minimal compared to the other test cases.

4.6.2 Results and discussion

S = Simultaneously
I = Individually

1. H1-H4 and H2-H5

Throughputs

Test	Interval	Recieved	Rate
H1-H4 (S)	0.0 - 25.1	25.15 MB	8.0 Mbps
H1-H4 (I)	0.0 - 25.1	54.93 MB	17.5 Mbps
H2-H5 (S)	0.0 - 25.6	28.66 MB	9.0 Mbps
H2-H5 (I)	0.0 - 25.0	55.69 MB	17.8 Mbps

The throughput for the simultaneous (S) tests for both H1-H4 and H2-H5 is much lower compared to when they are tested individually (I) as expected. The decrease in throughput can be attributed to the contention for network resources, as both pairs of hosts share the same routers (R1, R2, and R3) and links between them. When tested simultaneously, there is increased competition for these resources, leading to reduced bandwidth for both pairs of hosts. In contrast, when tested individually, each pair of hosts has exclusive access to the network resources, which allows for higher throughput.

Ping

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
H1-H4 (S)	24026ms	61.090	100.355	169.511	31.499 ms
H1-H4 (I)	24033ms	60.780	65.238	68.254	1.890 ms
H2-H5 (S)	24049ms	63.037	91.248	130.461	20.001 ms
H2-H5 (I)	24054ms	60.780	65.238	68.254	1.890 ms

When comparing the simultaneous (S) and individual (I) latency tests, there is a significant increase in the average RTT for both H1-H4 and H2-H5 when tested simultaneously. When tested simultaneously, there is increased competition for network resources, leading to higher latency as the packets experience more delays in transmission due to network congestion.

2. H1-H4, H2-H5 and H3-H6

Throughputs

Test	Interval	Recieved	Rate
H1-H4 (S)	0.0 - 25.0	20.61 MB	8.0 Mbps
H1-H4 (I)	0.0 - 25.1	54.93 MB	17.5 Mbps
H2-H5 (S)	0.0 - 25.3	16.72 MB	5.3 Mbps
H2-H5 (I)	0.0 - 25.0	55.69 MB	17.8 Mbps
H3-H6 (S)	0.0 - 25.1	21.37 MB	6.8 Mbps
H3-H6 (I)	0.0 – 25.2	38.17 MB	12.1 Mbps

When comparing the simultaneous (S) and individual (I) throughput tests involving H1-H4, H2-H5, and H3-H6, we can observe an even larger decrease in throughput for all pairs of hosts when tested simultaneously compared to test 1. This is due to the fact that all three pairs of hosts share the same routers (R1, R2, R3), links and switches between them.

Ping

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
H1-H4 (S)	24044ms	62.176	91.626	158.954	29.990 ms
H1-H4 (I)	24033ms	60.780	65.238	68.254	1.890 ms
H2-H5 (S)	24044ms	61.515	102.675	208.652	40.723 ms
H2-H5 (I)	24054ms	60.780	65.238	68.254	1.890 ms
H3-H6 (S)	24051ms	65.356	93.713	154.303	26.109 ms
H3-H6 (I)	24051ms	63.860	66.047	71.568	1.660 ms

We can also observe we can observe a significant increase in average RTT for all pairs of hosts when tested simultaneously.

3. H1-H4 and H1-H9

Throughputs

Test	Interval	Recieved	Rate
H1-H4 (S)	0.0 - 25.1	34.22 MB	10.8 Mbps
H1-H4 (I)	0.0 - 25.1	54.93 MB	17.5 Mbps
H7-H9 (S)	0.0 - 25.1	22.29 MB	7.1 Mbps
H7-H9 (I)	0.0 - 25.2	41.21 MB	13.1 Mbps

For H1-H4, the simultaneous throughput is 10.8 Mbps, which is lower than the 17.5 Mbps achieved when tested individually. Similarly, for H7-H9, the throughput during simultaneous testing is 7.1 Mbps, compared to 13.1 Mbps when tested individually.

H1-H4 and H7-H9 are partially separate. Since the paths do not entirely overlap, the shared network resources (routers and links) are not as heavily congested, which allows for better throughput compared to the cases where the paths have more overlap.

Ping

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
H1-H4 (S)	24038ms	66.034	96.674	156.584	25.554 ms
H1-H4 (I)	24033ms	60.780	65.238	68.254	1.890 ms
H7-H9 (S)	24046ms	62.448	87.989	146.277	21.684 ms
H7-H9 (I)	24048ms	62.433	66.870	69.473	1.625 ms

The average RTT increases significantly when tested simultaneously with H7-H9 (96.674 ms) compared to when tested individually (65.238 ms). Similarly, the average RTT increases when tested simultaneously with H1-H4 (87.989 ms) compared to when tested individually (66.870 ms).

These results indicate that the latency is affected when the paths are tested simultaneously, even though they are partially separate. The increase in latency can be attributed to the additional processing and queueing delays at the shared routers and links along the paths when multiple connections are active at the same time. This demonstrates that even partially separate paths can still experience performance degradation when multiple connections are active simultaneously.

4. H1-H4 and H8-H9

Throughputs

Test	Interval	Recieved	Rate
H1-H4 (S)	0.0 - 25.1	51.35 MB	16.3 Mbps
H1-H4 (I)	0.0 - 25.1	54.93 MB	17.5 Mbps
H8-H9 (S)	0.0 - 25.6	41.20 MB	13.1 Mbps
H8-H9 (I)	0.0 - 25.1	50.89 MB	16.2 Mbps

If we look at the results, when H1-H4 and H8-H9 connections are tested simultaneously, there is a decrease in throughput for both connections compared to when they are tested individually. The impact on throughput and latency is less pronounced compared to the first two tests. This is consistent with the fact that these two communication paths share only the R3 router, which causes some contention for network resources but not as much as in the first two tests where there were more shared resources involved.

Ping

Test	Time	Min (RTT)	Avg (RTT)	Max (RTT)	Mdev (RTT)
H1-H4 (S)	24043ms	61.157	87.305	138.233	23.016 ms
H1-H4 (I)	24033ms	60.780	65.238	68.254	1.890 ms
H8-H9 (S)	24054ms	21.457	29.529	55.283	8.064 ms
H8-H9 (I)	24047ms	20.642	22.955	27.294	1.402 ms

The ping results for the H1-H4 and H8-H9 test show a similar trend to the throughput results, with a slight increase in latency when the tests are run simultaneously compared to when they are run individually. The increase in latency and the decrease in throughput observed are relatively minor compared to the impact seen in the first two test cases.

4.7 Test case 5: effects of parallel connections

4.7.1 Introduction

Test case 5 involves testing the effects of parallel connections between three hosts (h1, h2, and h3) connected to R1 and three hosts (h4, h5, and h6) connected to R3. The objective is to measure the throughput when the hosts are communicating simultaneously using simpleperf. H1 will open two parallel connections to communicate with h4, while h2 and h3 will each have one connection. The test will run for a total duration of 25 seconds.

4.7.2 Results and discussion

H1-H4 1 and 2 are parallel connections.

Throughputs

Test	Interval	Recieved	Rate
H1-H4 1 (S)	0.0 - 25.1	13.32 MB	4.3 Mbps
H1-H4 2 (S)	0.0 - 25.8	14.81 MB	4.6 Mbps
H1-H4 1 (I)	0.0 - 25.0	29.61 MB	9.5 Mbps
H1-H4 2 (I)	0.0 - 25.4	20.42 MB	6.4 Mbps
H2-H5 (S)	0.0 - 25.0	13.13 MB	4.2 Mbps
H2-H5 (I)	0.0 - 25.0	55.69 MB	17.8 Mbps
H3-H6 (S)	0.0 - 25.0	13.13 MB	4.2 Mbps
H3-H6 (I)	0.0 - 25.2	38.17 MB	12.1 Mbps

Comparing the simultaneous and individual throughput results, we can see a decrease in throughput when multiple communications occur at the same time:

For H1-H4, the throughput drops from 9.5 Mbps (individual) to 4.3 Mbps (simultaneous) for the first connection and from 6.4 Mbps (individual) to 4.6 Mbps (simultaneous) for the second connection.

For H2-H5, the throughput decreases from 17.8 Mbps (individual) to 4.2 Mbps (simultaneous).

For H3-H6, the throughput reduces from 12.1 Mbps (individual) to 4.2 Mbps (simultaneous).

The topology plays a significant role in the results. H1-H4, H2-H5, and H3-H6 share common network resources such as links, routers and switches. More specifically, all three pairs share switches, routers R1, R2, and R3 and the links connecting them. This shared infrastructure leads to contention and competition for the limited available network resources when the hosts communicate simultaneously.

As a result, the throughput for each pair drops significantly during simultaneous communication compared to when they communicate individually. This highlights the importance of understanding the network topology and its impact on performance when multiple connections occur simultaneously, especially when they share common resources in the network.

5 Conclusions

5.1 Result summary

A summary of the results:

- Test case 1 showed that the bandwidth and transfer rate for different connections can be accurately measured using iPerf. It was observed that the transfer rate decreases when more connections are added to the network. Additionally, the jitter values increased with the number of connections, which suggests that the network experienced increased delays and fluctuations in data transmission.
- Test case 2 measured the latency and bandwidth of individual links between routers in the network. The results showed that L3 had a significantly higher latency than L1 and L2, which was likely due to various factors such as network congestion, packet loss, and jitter.
- Test case 3 focused on measuring the latency and throughput for specific paths between hosts in the network. The results showed that the latency and throughput varied depending on the path being taken, with H1-H4 having the highest throughput at 15.5 Mbps and H1-H9 having the lowest at 9.0 Mbps.
- Test case 4 tested the effects of multiplexing on network performance. It was observed that adding more connections led to a decrease in throughput and an increase in latency, as expected.
- Finally, test case 5 examined the effects of parallel connections on network performance. It was observed that using multiple parallel connections led to an increase in throughput, particularly for H1-H4.

In conclusion, the tests conducted on this network topology provided valuable insights into the impact of shared routers, links and parallel connections on throughput and latency. The results demonstrated that when communication paths share routers and links, the throughput and latency are significantly affected due to the contention for network resources. For instance, in test case 4 task 2, H1-H4, H2-H5 and H3-H6 all shared the same general path with the switches S1 and S2, routers R1, R2, and R3 and the links L1 and L2, which led to a noticeable drop in throughput and an increase in latency.

However, when communication paths shared fewer routers and links or had partially separate paths, the impact on throughput and latency was less pronounced, as observed in test case 4 (H1-H4 and H8-H9). This suggests that network design and management should carefully consider the distribution of traffic across routers and links to minimize contention for network resources and maintain optimal performance.

Furthermore, the presence of parallel connections, as examined in test case 5, also influenced throughput and latency, although the effects varied depending on the specific configuration. This highlights the importance of understanding the network topology and traffic patterns when evaluating and optimizing network performance.

Overall, these tests provide important insights into the performance of the given network topology. It was observed that factors such as the number of connections, path being taken, and the use of parallel connections and multiplexing can all have significant effects on network performance. By considering these factors, network administrators can achieve more efficient utilization of network resources and ensure optimal throughput and latency for various communication paths.

5.2 Discussion and conclusion

It is important to note that the tests performed in this project had some limitations. For example, the tests were conducted in a controlled environment using virtual machines, which may not accurately reflect real-world network conditions. Additionally, the tests only measured performance under specific conditions and may not be representative of all possible network scenarios. There may be limitations due to the fact that it was only performed on a specific network topology and the use of specific tools (IPerf, ping and simpleperf) to measure latency and throughput. The results may vary in different network topologies or under different test conditions, and the conclusions may not be universally applicable.

Despite these limitations, the results of this project can be useful for understanding network performance and identifying potential areas for improvement. By analyzing network performance in a controlled environment, it is possible to gain important insights into the factors that influence network performance and make informed decisions to optimize network performance in real-world scenarios.

In conclusion, working on this project has provided a valuable opportunity to gain hands-on experience in python coding, network testing and analysis. The project has taught me the importance of considering various factors such as network topology, bandwidth, latency, jitter, and multiplexing, and their impact on network performance. It has also allowed me to practice using various network testing tools such as ping and simpleperf and learn how to interpret their results. This project has highlighted the significance of testing and analyzing network performance in various scenarios to optimize network performance and prevent issues such as congestion or packet loss. Overall, the project has been a valuable learning experience in network testing and analysis, and I am grateful for the opportunity to work on it.

6 References

6.1 Litterature

Hardin, B., Comer, D., & Rastegarnia, A. (2018). *On the unreliability of network simulation results from Mininet and iPerf.* Journal of Communications and Networks, 20(6), 527-536. doi: 10.1109/JCN.2018.000086 https://www.researchgate.net/profile/Benjamin-Hardin-4/publication/369051833_On_the_Unreliability_of_Network_Simulation_Results_FROM_Mininet_and_iPerf/links/641b3a4092cfd54f84206698/On-the-Unreliability-of-Network-Simulation-Results-FROM-Mininet-and-iPerf.pdf

Kolahi S. S., Narayan, S., Nguyen, D. & Sunarto, Y. (2011) *Performance Monitoring of Various Network Traffic Generators.* International Journal of Computer Applications, 145(11), 15-20. doi: 10.5120/ijca2016911314. https://www.semanticscholar.org/paper/Performance-Monitoring-of-Various-Network-Traffic-Kolahi-Narayan/f150d190aaee12237aebeb594e0aac2f3b06cb63

6.2 Images/Illustrations

I, Safiqul. (2023) *topo-with-ips.*

https://oslomet.instructure.com/courses/25246/files/3164167?module_item_id=535140

6.3 Tools

VirtualBox: Oracle. (2023). Oracle VM VirtualBox. https://www.virtualbox.org/

Mininet: Mininet Project. (2022). Mininet. http://mininet.org/

Xterm: The X.Org Foundation. (2014). Xterm. https://invisible-island.net/xterm/

IPerf: The IPerf Team. (2016). IPerf - The TCP, UDP and SCTP network bandwidth measurement tool. https://iperf.fr/