APRIL 17, 2023

PORTFOLIO 1

PORTFOLIO 1

LEON TONGFAI LAM – S352005 - 236

Oslo Metropolitan University

1 Introduction

In our modern digital age, computer networks have become a central piece to modern business and everyday life. Organizations rely on networks for communication, data transferring and much more. However, to ensure an optimal network, it is essential to understand how a network is structured, how packets travel, and the rate information can be transmitted and received.

This report aims to provide a comprehensive overview of a specific network topology with multiple tests in both latency and throughput. Results from several methods of testing latency and throughput will be displayed and further understanding of the results will be covered correspondingly.

In multiple test cases, we will explore network latency with the use of a network utility called "Ping". We will define latency, discuss factors that contribute to it, and explain how one can measure and perhaps reduce latency on a network system.

Next, we will delve into network throughput and its importance in ensuring network performance. Additionally, discuss factors that affect its performance and explore different methods using both "Iperf" and our simplified version "Simpleperf".

When it comes to measuring and testing network systems, it's essential to create a testing environment that resembles a real-life scenario. However, in our case we are limited to a specific small-scale network topology, to allow testing and measuring on our own systems on hand. These limitations can impact the results of network performance and should not be used to represent a real-world scenario.

By the end of this report, one should have a further understanding of how a network is structured, latency, throughput, and their importance in network performance. They will also gain insight into multiple practices for measuring and testing, perhaps a few measures to optimize network performance.

2 Simpleperf

Simpleperf is a simplified tool based on "Iperf" – an open-source command-line tool that allows users to measure network performance both with TCP and UDP protocols. Our tool allows users to measure throughput by sending data packets between a server and a client with TCP protocols.

To measure throughput, the user would first install the tool on the system and invoke the tool as a server on the recipient server. The transmitting server or system would then invoke simpleperf as a client and attempt to connect to the server, allowing the two hosts to communicate through data-packets of the size 1000 bits.

Simpleperf provides several options when invoking as server or client. As a server, it allows the user to select which host to bind to, the port to be listened on and in which format you wish the total data sent is displayed in. On the other hand, the client includes setting which address and port it should connect to, the format of the output data, total duration or maximum data that should be generated, in however many seconds intervals it should print statistics and number of parallel connections to the server.

More on how the tool is configured and ran when tested can be found in the README.txt file provided.

3 Experimental setup

In the provided topology (Safiqul) portfolio-topolgy.py, each host is connected to multiple other routers, creating a decentralized network with a combination of paths for data to travel. This topology is straight-forward and allows for testing simpleperf and allowing users to easily identify potential bottlenecks and other issues that may impact network performance.

In a virtual network, devices can be simulated using virtual clients and servers. Each device can be measured with the simpleperf, and data can be transmitted between devices using TCP and a variety of testing configurations. The tool records various metrics, more on that later.

By using a virtual network topology that is decentralized, testers can evaluate how well the network measurement tool performs under an optimized network condition, one with little traffic loads, low to no congestion and often without failures. This can help identify areas where the tool may be imprecise or incorrect, allowing improvements to enhance the tool's accuracy and performance.

4 Performance evaluations

4.1 Network tools

Iperf is a command-line tool used to measure network performance. In our case, it is used and tested with UDP protocols to output the networks throughput. It also allows

users to test other aspects of network performance, such as latency and packet loss. The tool also supports TCP protocols and provides various options to configure how data is transferred, including settings for packet size, number of packets, and the duration of tests. (Dugan, Elliott, Mah, Poskanzer, & Prabhu)

Ping is a command-line network tool that sends echo requests to a target host or IP to measure availability and the round-trip time (RTT) of the network. It works by sending the target host an ICMP (Internet Control Message Protocol) echo request and receiving an ICMP echo back from the server. By measuring the time between the transmission and reception, ping calculates and outputs the RTT of the connection. More on how ping can be initiated in the README.txt.

Ifconfig is a command-line tool used in Linux to allow users to configure and display network interfaces on a system. When the command is executed without any options, it displays the information for all network interfaces on the system. This tool will be used to display each individual node of the topology. (IBM, 2023)

4.2 Performance metrics

Most importantly from Simpleperf and Iperf, throughput which refers to the amount of data transmitted over the network in a given period of time.

Latency refers to the time it takes for a packet of data to travel from a start-point to an endpoint. Often referred to in milliseconds (ms).

RTT refers to the time it takes for a packet of data to travel between two points and back. Also measured in milliseconds (ms).

4.3 Test case 1: measuring bandwidth with iperf in UDP mode

4.3.1 Results

With the use of 'IPerf' a command line tool to test network throughput between two network hosts, we can measure the bandwidth of three separate host to client connections.
The measurements are generated through UDP load (traffic) from a host to a client, and in return, gives results regarding maximum data transfer rate from client or host, total data sent and loss.

Between client-server pair h1 – h4:

Client bandwidth – 29.4 Mbits/sec

Server bandwidth – 28.1 Mbits/sec

Between h1 – h9:

Client bandwidth – 21.0 Mbits/sec

Server bandwidth – 18.9 Mbits/sec

Between h7 – h9:

Client bandwidth – 21.0 Mbits/sec

Server bandwidth – 18.9 Mbits/sec

4.3.2 Discussion

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

Choosing the maximum bandwidth according to the capacity of the bottleneck link passed, will ensure an accurate measurement of the maximum bandwidth that the link(s) can support. Though, it is possible to achieve a lower throughput if the bottleneck link is congested and if there are other interrupting network factors in play. Choosing a client rate higher than what is supported, will only lead to congestion, higher worktime, and more loss of data. (Safiqul)

(Expected rates are found on Canvas, "portfolio-guidelines.pdf" or by looking at the topology (portfolio-topology.py)

Between the pair h1 – h4, choosing a rate of 28M results in a server bandwidth of 28.1 Mbits/sec. The bottleneck link is L2 with an expected rate of 30 Mbits/sec (Safiqul). The outputted server rate recorded is slightly lower than the expected rate, to be expected, taken interference and other network variables into consideration.

Between h1 – h9, choosing a rate of 20M gives a server bandwidth of 18.9 Mbits/sec. The connection passes all links, including the bottleneck link L3 with an expected bandwidth of 20 Mbits/sec. The actual outputted rate is slightly lower, and we carry the same assumptions as the previous measurement.

Between h7 – h9, the same bottleneck link is passed, and with a client rate of 20M and a server rate of 18.9 Mbits/sec. The results are expected to be the

same as h1 – h9, since the same bottleneck link is passed, and the expected maximum rate is unaffected.

4.3.2.2 *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.*

Identify the optimal packet size and test duration for achieving highest bandwidth. Do this by performing multiple sessions between the same endpoints whilst varying the parameters. In addition, perform multiple tests under different network conditions, as various network factors such as network congestion, network traffic etc. can affect the measurements.

Analyze the bandwidth measurements from both client and server, and packet loss to get a better understanding of the network performance.

In my case, I would start a test with a lower (than usual) rate to ensure that the network doesn't become congested or negatively impact other network traffic. Steadily increasing the rate, until the test results level out or peak at a certain rate, doing this during periods of low network utilization, minimizing any potential interference. In the end, analyzing the results and performing multiple tests with what is considered optimal, ensuring accurate measurements and as low as possible impact on network performance.

4.4 Test case 2: link latency and throughput

4.4.1 Results

In this task, data such as RTT and bandwidth are measured between three individual links between routers. Ping, a networking utility is used to test reachability to a target host, while outputting a measurement of round-trip time (RTT). Simpleperf, a network throughput measurement tool to send and receive packets between a client and server using sockets with "Transmission Control Protocol (TCP)". (Gillis, 2021)

Average RTT of L1-L3:

L1 (r1 – r2): 25.5 ms
L2 (r2 – r3): 45.7 ms
L3 (r3 – r4): 23.0 ms

Throughput of L1-L3:

L1 (r1 – r2): 21.15 Mbps
L2 (r2 – r3): 12.94 Mbps
L3 (r3 – r4): 12.89 Mbps

4.4.2 Discussion

According to the guidelines (Safiqul) or by looking at the topology, expected latency between the individual links display the time (ms) it takes from start-point to end-point. Expected RTT in this case would then be start - end, then end - start, totaling ca. doubled the time provided.

Throughput between individual links is also provided, though in this case our results are heavily impacted by unknown variables. The same tests using simpleperf.py on another computer shows results much closer to expected throughput proving our theory.

L1:

Expected latency is 10 ms and expected RTT is 20 ms, if not more taking network variables into consideration. Results show an average RTT of 25.5 ms, and the discrepancy could be caused by the distance between hosts and the processing delay from transmission and reception.

Expected maximum throughput is 40 Mbps, if not more considering various variables. Through testing (on my computer), we are provided a result of 21.15 Mbps, making us believe that the testing environment and network is heavily impacted by hardware limitations and system configuration. On another computer, we are provided with a result of ~38 Mbps, which is expected considering the network is not congested nor impacted by network traffic.

L2:

Expected latency is 20 ms and expected RTT is 40 ms or slightly more. The average RTT through testing is 45.7 ms, with close to the same margin of error of L1.

Expected maximum throughput is 30 Mbps, and result show 12.94 Mbps. As mentioned above, on another system results show much higher rate (slightly lower than expected throughput), which is to be expected taking network variables into consideration.

L3:

Expected latency is 10 ms and expected RTT is 20 ms or more. The average RTT from testing is 23.0 ms, faster than the other tests, but within margin of error.

Expected maximum throughput is 20 Mbps, and result show a throughput of 12.89 Mbps. From another source, we are provided a result of 18.9 Mbps (or slightly under 20 Mbps), which is expected on a network with distance between hosts and potentially multiple sources of interference (traffic), creating processing delay.

In conclusion, the expected round-trip time (RTT) and throughput in practice is often or always weaker than provided, caused by several factors.

- Network congestion, increased travel time of packet.
- Distance between hosts, requiring packets to travel a longer distance.
- Network latency, affected by the quality of the connection, number of connections from client to host, connection protocol, etc.
- Processing delay, time it takes to process data/packets, affected by how busy a host is or how much processing load present.
- Hardware limitation or difference, system with different configuration and computing power, resulting in different processing time, latency, etc.

In our case, the other system presents results that are much closer to what is expected, whilst the system which we got the actual results from was outputting unoptimized tests results.

4.5 Test case 3: path Latency and throughput

4.5.1 Results

In this case, RTT and throughput is measured between two hosts, creating connections across multiple links. Tools used are – Ping to measure RTT (ms) and simpleperf to measure throughput (Mbps).

Average RTT through paths:

H1 – h4: 64.6 ms
H1 – h9: 87.0 ms
H7 – h9: 65.3 ms

Throughput through paths:

H1 – h4: 14.84 Mbps
H1 – h9: 11.85 Mbps
H7 – h9: 12.44 Mbps

4.5.2 Discussion

By looking at the topology and reading latency and bandwidth of each individual link, it is possible to determine what the expected results for each path before testing. RTT is determined by the time it takes to send data through the path across link(s), and all the way back again. Throughput is determined by whichever link passed with the lowest bandwidth, also known as the "bottleneck link".

In the case of h1 – h4, the links L1 and L2 are passed through to test both latency and throughput. Expected latency totals up to 30 ms and expected RTT is thus 60 ms or more. Results show a little above 60 ms (64.6), which is to be expected when considering the time to connect through links, processing time etc. Expected throughput is based on the bottleneck link with the lowest maximum bandwidth L2 (30 Mbps), and the test outputs 14.84 Mbps which is inaccurate considering the network is not experiencing traffic nor congestion when running simpleperf. On another computer, we would get results showing throughput around 28 Mbps, which is a lot more accurate when taking several network variables into consideration.

H1 – h9, all links are passed when testing latency and throughput. Expected latency totals to 40 ms and expected RTT is thus 80 ms or more. Results show just more than 80 ms (87.0 ms), a result expected considering the fact that more links are to be connected when testing with the "Ping" tool. Expected throughput is based on the bottleneck link L3 with a rate of 20 Mbps, though results show 11.85 Mbps, a much lower rate to be considered accurate. On another system, we received throughputs around 18 Mbps, results considered more accurate and valid the report.

H7 – h9, where links L2 and L3 are passed through. Expected latency is 30 ms and expected RTT is 60 ms or more. Tests gives an average RTT of 65.3 ms, which is to be expected like the tests previously. Expected throughput is based on L3 with a rate of 20 Mbps, though results output 12.44 Mbps, likely due to less connections between links in the path given. Another test on a different system shows results around 19 Mbps, which are more likely because of how little interference there should be on the network topology we are using.

In conclusion, expected RTT can be determined by understanding the topology and each links that are interconnected when testing paths. Results that output RTT with slightly higher time than what is expected, is most likely due to the time it takes to connect to each link, etc. Expected throughput is determined by the bottleneck link's maximum throughput rate and often returns results slightly lower, likely due to the time to connect, etc.

4.6 Test case 4: effects of multiplexing and latency

4.6.1 Results

In this case, the method multiplexing is performed with tests of latency and throughput. The method to perform multiplexing is in our case done by initiating the necessary tests as fast as possible, to let them process on top of one communication channel. Expected results will differentiate at the beginning and at the end of each test, since they are out of sync, start and end at different times.

Average RTT:

H1 – h4	h2 – h5
64.868 ms	66.126 ms

H1 – h4	h2 – h5	h3 – h6
70.022 ms	67.041 ms	65.829 ms

H1 – h4	h7 – h9
66.323 ms	66.022 ms

H1 – h4	h8 – h9
68.117	23.957 ms

Throughput:

H1 – h4	h2 – h5	Sum
7.63 Mbps	6.51 Mbps	14.14 Mbps

H1 – h4	h2 – h5	h3 – h6	Sum
5.08 Mbps	4.29 Mbps	4.59 Mbps	13.96 Mbps

H1 – h4	h7 – h9	Sum
9.99 Mbps	5.14 Mbps	15.13 Mbps

H1 – h4	h8 – h9	Sum
20.06 Mbps	15.65 Mbps	Irrelevant

4.6.2 Discussion

To further understand the results from this case of testing, we must get an understanding the effects of multiplexing. The method multiplexing is as mentioned, a way of sending multiple steams of data over a communication link at roughly the same time. The individual network signals are transmitted (in some cases) through a shared medium (links) and outputted individually for use by other operations, in this case, to be analyzed. (Sheldon & Burke, 2021)

The way of determining the expected latency remains the same. Testing using "Ping" with multiple streams of data only puts a load of 64 Bytes * n (number of streams) every transmission. If you compare the total load with the maximum rate of whichever link passed, one can easily tell that the network will not be congested nor interfered by the traffic. Therefore, results of latency are not noticeably affected by multiplexing and remain the same as if the tests were performed without multiple streams.

On the other hand, determining throughput is based on how many streams are sharing a bottleneck link.

h1 – h4 and h2 – h5:

In our first test, the pairs are simultaneously measured using simpleperf, forcing the two network signals to divide the maximum bandwidth of the bottleneck link L2. The maximum bandwidth is 30 Mbps, meaning that the throughput of both pairs has an expected rate of less than 30 Mbps.

In our case, the system used is heavily limited, only allowing both streams to output 14.14 Mbps in total. On another system the same test was performed and gave us a total throughput of roughly 28 Mbps unevenly shared between the two streams.

The uneven distribution is caused by how one of the streams was executed before the other, allowing it more bandwidth until the latter connected.

In some cases, the combined throughput would be larger than the expected throughput. The reason behind this is because if one stream finished significantly earlier than the other(s), the remaining stream(s) would be allocated rest of the resources from the less congested bottleneck link.

H1 – h4, h2 – h5 and h3 – h6:

In this test, three network signals share a single path and are forced to divide the maximum rate of L2. The expected throughput would be all three streams combined and less than 30 Mbps, unevenly distributed based on whichever order the pairs were initiated and the time between each execution.

In our case, the total available bandwidth was around 14 Mbps (based on the test before) and we would get a combined throughput of 13.96 Mbps. On another system, combined throughput outputs around 28 Mbps which is to be expected in an optimized environment, but quite unlikely on a realistic network with traffic and interference.

H1 – h4 and h7 – h9

Two network signals share different paths, but "collide" when connecting through L2. Expected throughput is based on L2's maximum bandwidth, forcing the two network signals to divide on a rate of 30 Mbps.

In our case, the combined throughput is 15.13 Mbps which is still significantly lower than the expected throughput, but higher than the first test, likely due to the fact the signals only needed to share one link before going through the bottleneck link slowing the rate down to what is left available.

H1 – h4 and h8 – h9

Two network signals that do not need to share any links' bandwidth. The first pair has an expected throughput 30 Mbps and the latter 20 Mbps. This is due to the paths not overlapping removing the need to share a bottleneck link's bandwidth.

In our case, the first pair has a throughput of 20.06 Mbps and the latter 15.65 Mbps. Though the results should not affect each other, it is difficult to compare the actual results to what is expected since they do not even correlate to the throughput each link.

On another system, the first pair has throughput of around 29 Mbps and the latter 18 Mbps. Results considered valid even though they cause slight interference by transmitting to r3 at the same time.

4.7 Test case 5: effects of parallel connections

4.7.1 Results

In this case, the measurement of throughput is performed on three pairs of hosts and four streams of simpleperf. All the hosts plus an additional connection from R1 are to be connected to all the corresponding hosts connected to R3, while transmitting and receiving packets of data to be measured. The additional connection from R1 is executed by invoking H1 with argument –parallel 2 or -p 2.

Throughput (all initiated roughly simultaneously):

H1 – h4 (-p)	H1 – h4 (-p)	H2 – h5	H3 – h6	Sum
3.52 Mbps	2.85 Mbps	5.41 Mbps	2.65 Mbps	14.43 Mbps

4.7.2 Discussion

The way of determining the expected throughput of all connections simultaneously connected is a combination of understanding the results from test case 3 & 4. Firstly, the expected throughput of a single network signal on a path is based on the bottleneck link's maximum bandwidth. Secondly, the expected throughput of multiple network signals on a single channel is based on the bottleneck link's max bandwidth (unevenly) divided on the amount of network signals.

In this case, we are invoking the pair h1 – h4 with two parallel connections in addition to the pairs h2 – h4 and h3 – h6, all on one single channel across links L1 and L2. Combined, we can determine the expected throughput being the bottleneck link's max bandwidth divided by 4 (unevenly).

The maximum bandwidth being around 14 Mbps based on results from test case 2 and from previous tests between h1 – h4. The expected throughput based on that, is around 14 Mbps divided by 4 (unevenly). What we get in return by testing totals to 14.43 Mbps, which is slightly more than expected, likely due to a change in the testing environment.

Though on another system, the expected throughput is based on L2's maximum bandwidth being 30 Mbps, divided by 4. The result from testing returns a combined sum of 29 Mbps, a more valid answer since the testing environment is in an optimal state allowing for "multiplexing".

5 Conclusions

Understanding of networks and topology, latency and throughput are important because it helps us to understand how networks work and how we can as developers optimize and identify shortcomings for better performance.

In addition, a network measuring tool is essential for monitoring and evaluating network performance, allowing us to measure key metrics such as throughput, latency and so on. It also helps user identify potential performance issues, troubleshoot, and ensuring the network is meeting the needs of applications and users.

Overall, a deeper understanding of the modern infrastructure is essential for building and maintaining a network that supports the needs of modern applications and users.

6 References (Optional)

Dugan, J., Elliott, S., Mah, B., Poskanzer, J., & Prabhu, K. (n.d.). *Iperf.fr.* Retrieved from Iperf: https://iperf.fr/

Gillis, A. S. (2021, August). *techtarget.* Retrieved from network topology: https://www.techtarget.com/searchnetworking/definition/network-topology#:~:text=A%20network%20topology%20is%20the,often%20represented%20as%20a%20graph.

Islam, S. (2023, Mars 16). *Github.* Retrieved from portfolio-topology: https://github.com/safiqul/2410/commits/main/portfolio-topology/portfolio-topology.py

JavaTPoint. (n.d.). *Javatpoint.* Retrieved from Ping: https://www.javatpoint.com/ping-command-in-linux

Safiqul, I. (n.d.). *Canvas.* Retrieved from DATA2410 - portfolio-guidelines.pdf: https://oslomet.instructure.com/courses/25246/files/3153005?module_item_id=533441

Sheldon, R., & Burke, J. (2021, August). *Techtarget.* Retrieved from multiplexing.