APRIL 17, 2023

PORTOFOLIO

SIMPLEPERF IMPLEMENTATION

MOHANAD NAFI AL-FALAH
S362092
Oslo Metropolitan University

1 Introduction

The term "bandwidth" describes the most data that may be sent via a communication link in a specific length of time. It is a crucial idea in networking and a major determinant of how well a network performs.

For modern networks, where high-speed connections are necessary to serve data-intensive applications like video streaming, cloud computing, and virtual reality, the issue of maintaining adequate bandwidth is crucial. Insufficient bandwidth can cause connections to disconnect, applications to run poorly, and user unhappiness, which can result in lost productivity.

Iperf, a widely used tool for assessing network throughput, is one of several tools and methods that have been developed to measure and enhance network bandwidth. Traffic shaping, QoS methods, and bandwidth shaping are further pertinent works in this field.

The approach for solving the bandwidth issue include streamlining network architecture and setting up network hardware to optimize available bandwidth while reducing latency and packet loss. To achieve the best possible network performance, we also take into account variables like network structure, routing, and congestion control.

Our method has the drawback of requiring specialized technical knowledge and skills, making it potentially inappropriate for non-technical users. Our strategy's results, however, have the potential to significantly boost network performance, user satisfaction, and overall productivity.

The rest of this document will give an overview of bandwidth ideas, examine several methods for increasing bandwidth, talk about the drawbacks and trade-offs of various strategies, and offer suggestions for increasing network bandwidth in different contexts.

2 Simpleperf

I will explain this code step by step, I mean I will explain the task of each function in this code in a brief way so that it becomes easy to read and understand for everyone. First, I have three basic functions in this code: 1. handle_client 2. the server function 3. the client function I will explain about each of these functions.

The function handle_client, which accepts three arguments: a client socket, an address, and a multiplier. The client transmits the string "BYE" before the function stops accepting data from the client in a loop. The information is calculated and printed to the console, together with the amount of data received, the length of the connection, the transfer size, and the transfer rate. Additionally, it cuts off the client socket connection and returns an acknowledgment message to the client. Additionally, the script imports a number of modules, such as (socket) for creating and using sockets, (threading) for managing multiple threads, (time) for obtaining the time and measuring durations, (argparse) for processing command-line arguments, (re) for regular expression operations, and (sys) for interacting with the system. The script also specifies a constant variable (BUFFER_SIZE) with a value of 1000, which establishes the most data the client can send at once.

The size of the buffer used to receive data from the client is determined by the (BUFFER_SIZE) variable. The size of the buffer used to receive data from the client is set in this code by the constant BUFFER_SIZE. Data is often received across a socket connection in chunks, and the size of these chunks might have an impact on the application's speed. Setting a larger buffer size will allow the program to receive bigger data chunks at once, which will enhance the application's overall

performance by lowering the number of times the (recv) method on the socket object is called. However, if several client connections are being handled at once, making the buffer size too big may result in excessive memory utilization. A buffer size of 1000 bytes, which is a comparatively tiny buffer size, is utilized in this function. If the data being transferred is little or the connection has a limited bandwidth, this can be the best option. However, the right buffer size will vary depending on the use case and should be determined by the demands of the program.

The server function accepts the argument args as a parameter. The host, port, and format values are first taken out of the args object by the function. It determines if the format is legitimate and computes a multiplier using the format value. The function attempts to bind the server's newly created socket object to the provided host and port. The function outputs an error message and exits if the bind action fails. The function sets the server to listen for incoming connections and outputs a message indicating that the server is listening if the bind procedure is successful. After then, the function enters an indefinite loop where it keeps accepting new connections and starting a new thread to handle each one. A new thread is created to handle the client connection using the handle_client function when a client connects, and the function prints a message containing the client's address. In order for handle_client to appropriately calculate the transfer size and transfer rate depending on the provided format, the multiplier that was previously calculated is passed as an input. The server function, in general, configures a server to listen for incoming connections, accepts and manages connections using the handle_client function in separate threads, and computes the transfer size and transfer rate based on the supplied format.

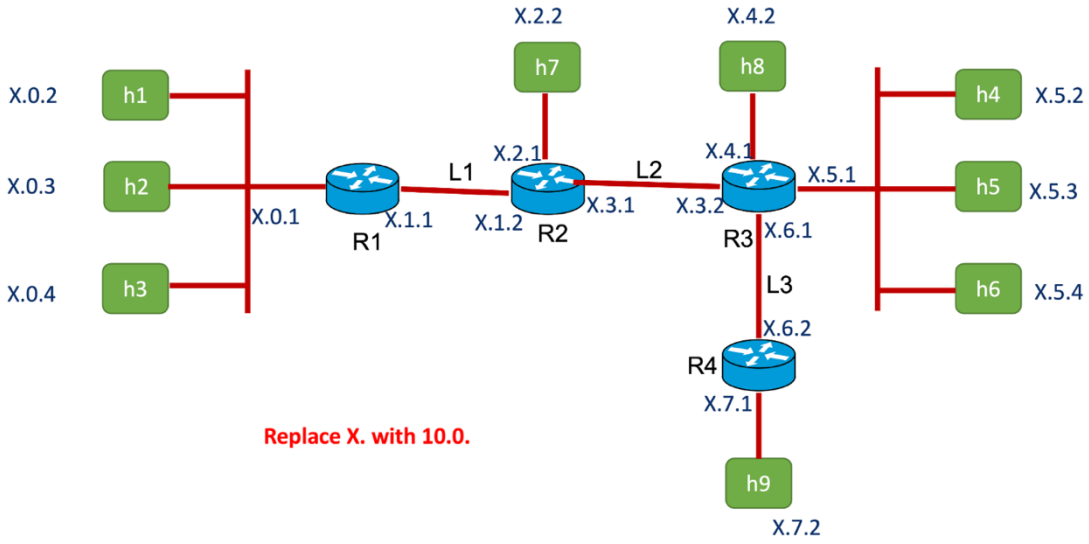
The client function accepts an args parameter with multiple properties, including serverip, port, format, time, and interval. The function's goal is to connect to a server at the IP address and port that are supplied by serverip and port, respectively, and transfer data to the server at a predetermined rate. 'B', 'KB', or 'MB' are the three possible values for the format property, which specifies the format of the data to be delivered. The format argument is first verified as valid by the function. If not, it prints a message with an error and exits the function. The format value is used to generate a multiplier that will be used to convert the data rate to bytes per second. The time and interval arguments, which describe the overall length of the data transfer and the interval at which data should be delivered, respectively, are then retrieved by the function from the args array. In addition to creating a socket and connecting to the server, it initializes variables for measuring the volume of data sent. The function ends by printing a message confirming the client's connection to the server. interval_time, is data transmission and periodic transfer statistics calculation are handled by this code block. If the interval_time parameter is not zero, the function enters a loop where it determines how many intervals there will be based on total_duration and interval_time, sends data for the duration of each interval, and computes transfer statistics for that interval. The loop outputs statistics for each interval and cumulative statistics for all intervals at the conclusion of the loop. The connection is then terminated by the client sending a "BYE" message to the server. The function delivers data and computes transfer statistics throughout the duration of the test if the interval_time argument is zero. In this instance, it prints test-wide statistics and disconnects from the server by sending a "BYE" message.

args.num, this function is in charge of computing the transfer statistics and delivering a certain quantity of data to a server. The entire transfer size is specified by the args.num parameter, which is then extracted and converted to bytes based on the supplied unit (e.g., KB, MB). After that, until the total number of bytes delivered matches the requested transfer size, the data is sent to the server in BUFFER_SIZE chunks. The client sends a 'BYE' message to signal that the transfer is finished after transferring all the data. The transfer statistics, which include the magnitude, rate, and duration of the transfer, are then computed.

This script uses sockets to construct a condensed version of the iperf tool. The script offers command line arguments for specifying numerous parameters such as the IP address, port number, duration of the test, number of parallel connections, transfer rate format, etc. It also supports both server and client modes. The parse_args() method is used by the main() function to parse the command line

arguments after setting up an argument parser to handle them. Then, if neither server nor client modes are given, server() is called for server mode and client() for client mode, respectively, with the parsed parameters. The script launches many threads for the client() function if the parallel parameter is supplied, each of which will use the same parsed arguments. The main() function is then called if the script is being executed directly (as opposed to being imported as a module).

3 Experimental setup



As I understood from the image and its code. I will explain a little bit about this. This is a topology for a computer network using the Mininet Python API. The topology of the network consists of several subnets that are linked to one another by routers. There are several subnets in the network topology, each of which includes hosts and switches. Routers are used to link the subnets together. A switch and a router (r1) are joined to three hosts (h1, h2, and h3). A connection connects each host to the switch. A connection connects the switch and the router. Two routers (r1 and r2) are linked to one another. Both r1 and r2 have IP addresses; each of them is assigned a unique one. The link between r1 and r2 has a 40 Mbps bandwidth, a 10 MS delay, and a 67-packet maximum queue size. A new host named (h7) is associated with r2. h7 has its own IP address. Two routers (r2 and r3) are linked to one another. R2 and R3 each have their own unique IP addresses. The link between r2 and r3 has a 30 Mbps bandwidth, a 20 MS delay, and a 100-packet maximum queue size. There is a new host named (h8) that has an IP address and is linked to the third IP-assigned interface of router r3 (r3-eth1). A switch is connected to three hosts (h4, h5, and h6). Each of the following has an IP address: h4, h5, and h6. A link with a bandwidth of 30 Mbps, a delay of 20 MS, and a maximum queue size of 100 packets connects the switch to r3. A new router node called r4 is made, connected to r3, and given an IP address using the r3-eth3 interface of r3. The IP address is also given to the r4 node. The link between r3 and r4 has a 20 Mbps bandwidth, a 10 MS delay, and a 33-packet maximum queue size. A new host node named h9 is created and assigned the IP address. A link is added between r4 and h9 using the r4-eth1 interface of r4 and assigning it the IP address.

4 Performance evaluations

4.1 Network tools

Both the network testing tools Iperf and Ping are used to assess network performance and identify connectivity problems.

Ping is a command-line tool that sends a target host or IP address an ICMP echo request message and then waits for an ICMP echo reply message. The "ping time" or "round-trip time," which is used as a gauge of network latency or delay, is the amount of time it takes for the request and reply to messages to travel back and forth. Ping can also be used to check the accessibility and responsiveness of a target host or IP address. When the ping is low, the response speed is higher, which means that the internet speed is faster. Pings speed is calculated in ms.

In contrast, Iperf is a network testing tool that calculates the highest possible throughput or bandwidth between two endpoints, often a client and a server. Iperf measures the amount of time it takes for a set amount of data to be sent from the client to the server. As a result, it can determine the bandwidth or the volume of data that can be transferred in each amount of time. Other network performance measures, such as packet loss, can be measured using Iperf.

Iperf is used to monitor network capacity and performance between two endpoints, whereas ping is used to measure network latency and establish whether a target host or IP address is reachable.

4.2 Performance metrics

The performance metrics that I use to evaluate my simple perf tool, it is latency, throughput, and bandwidth. Latency, throughput, and bandwidth are all important metrics that are used to measure the performance of communication networks, including the Internet. While they are related, they each measure different aspects of network performance. I will explain each one of them.

Latency: The time elapsed between the time a request is issued and the time a response is received is referred to as latency. Milliseconds (MS) are a common unit of measurement. Distance between sender and receiver, the number of network devices the request must transit through, and the processing speed of the devices themselves are some of the variables that affect latency. Applications that demand real-time data, such as video conferencing, online gaming, and phone communication, will respond more quickly and offer a better user experience on a network with low latency.

Throughput: The amount of data that may be transmitted via a network in a specific amount of time is referred to as throughput, on the other hand. Bytes per second (Bps) or bits per second (bps) are frequently used to quantify it. The amount of available bandwidth, the number of users using the network, and the effectiveness of the network protocols in use all have an impact on throughput. A network with high throughput will be capable of efficiently and swiftly transferring massive amounts of data.
Bandwidth: The maximum quantity of data that may be sent via a network in a specific amount of time is referred to as bandwidth. Bytes per second (Bps) or bits per second (bps) are frequently used to quantify it. The amount of network capacity that is available, the number of people sharing the network, and the topology of the network all have an impact on bandwidth. A network with plenty of bandwidth can handle a lot of traffic and handle huge amounts of data.

In summary, the difference between them is, bandwidth measures the maximum amount of data that can be transmitted over a network in each amount of time, and throughput measures the amount of data that can be transmitted over a network in each amount of time, and latency measures the delay between sending a request and receiving a response. Although they are related, they are separate measures that each contribute significantly to assessing network performance.

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

What I did in the first case, I made 3 separate iperf measurements UDP mode between h1 and h4, between h1 and h9, and between h7 and h9. So, in the first measurement between h1 and h4, h4 was a server and h1 was a client, I linked them using this command on the server iperf -s -u, and on the client, I used this command iperf -c 10.0.1 .2 -u -b 30M. (- u) are specifies UDP mode, and (-b) specifies the rate at which I want to send, for example, 30 Mbps here. I did the same thing between the rest measurements between h1- h9 and between h7 - h9. With iperf, we can do easy tests of the network performance between a client and a server; principally the throughput, but also other elements like for instance packet loss. The server side first operates in UDP mode, after which it is listening and prepared to accept any connections. I'll start working with the client side now. Using the iperf command and the -u flag to instruct it to use UDP, we run the iperf program to the same target. Additionally, we must select the application's sending speed while using UDP. When using TCP mode, the program merely transmits data to the transport protocol, which in this case will automatically govern the speed. In this instance, the UDP does not regulate the speed; instead, it sends as much data as is physically possible through the network or link. As a result, we must provide the client's transmission rate using the bandwidth option -b, followed by the rate in megabits per second. By default, the connection should complete in 10 seconds, after which the output appears.

4.3.1 Results

	Expected (Bandwidth)	Reality (Bandwidth)
h1-h4	30M - 0%	28M - 0%
h1-h9	20M - 0%	17M - 0%
h7-h9	20M - 0%	18M - 0%

4.3.2 Discussion

Explain your results (what you expected vs what you got)
I will explain the first result between h1 and h4, I can infer that the highest bandwidth I can acquire between h1 and h4 is 40MB as there is 40MB of available bandwidth between r1 and r2. Since I must pass through this link to get to h4, I must also take its bandwidth into account. Compared to the bandwidth between r1 and r2, the bandwidth between r2 and r3 is less, at 30MB. The bandwidth between h1 and h4 will therefore be restricted to 30MB. When I tried 30MB, I got more than 1%, and the packet loss and the packet loss cannot be more than 1%, it should be between (0 - 1%). So, I tried another number like 29MB also I got more than 1%, in the end, I tried 28 MB, so I got 0% and this is the correct answer. The same situation is in the other test, such as between h1-h9, where I should only use 20MB of the maximum bandwidth to achieve the greatest results with the least amount of logical loss. Since we only go through three routers in the other tests, going through four routers here could result in lower latency than in those measurements. I tried in the same way as the first measurement between h1 and h4, when I tried 20MB, I got more than 1%, and the packet loss and the packet loss cannot be more than 1%, so I tried 17MB and I got 0%. Also, in the same situation in the last test between h7 and h9, the maximum bandwidth is 20MB, I did also like the other tests I tried many numbers than a low of 20, but I got more than 1%, so in end, I tried 18, so I got 0%. So, to have the best UDP connection with as little packet loss as possible, I would mostly choose the values

I did to avoid having poor connection quality when transferring data. If I were asked to use iPerf in UDP mode to measure the bandwidth without any knowledge of the network topology, I would run experiments with various packet sizes and traffic rates to establish the maximum sustainable bandwidth using iPerf in UDP mode. The real available bandwidth may differ from the anticipated bandwidth, so it may not be completely accurate. In network circumstances that are continually changing, this strategy might not be feasible.

4.4 Test case 2: link latency and throughput

4.4.1 Results

Throughput	Bandwidth	Throughput I reality
L1 (r1-r2)	40Mbps	38.12 Mbps
L2 (r2-r3)	30Mbps	28.42 Mbps
L3 (r3-r4)	20Mbps	19.8 Mbps

Latency	Expected RTT (average)	RTT in reality
L1 (r1-r2)	10ms	22.6ms
L2 (r2-r3)	20ms	45.7ms
L3 (r3-r4)	10ms	23.2ms

4.4.2 Discussion

In this case I had to test the link between the four routers (r1, r2, r3 and r4) in the topology. We should establish a baseline expectation for each link's performance based on variables like link speed, network congestion, and other potential bottlenecks to compare the expected vs. actual performance. The distance between each link's two ends and any potential routing changes for the traffic should also be considered. Once we have established your baseline expectations, you can compare them with the actual results of your tests to identify any performance issues or discrepancies. For example, if the actual throughput is significantly lower than expected, it may indicate congestion or other network issues that need to be addressed. I can discuss about my results for Throughput: Since they are not the same thing, comparing bandwidth and throughput is not a perfect comparison that can be made. The greatest quantity of data that can be sent between two links is known as bandwidth. However, throughput is the actual amount of data that is delivered via a connection in a specific amount of time. In this case, we are measuring throughput for 25 seconds by using 25. The results of the throughput testing show that, despite the bandwidth occasionally being significantly higher than the throughput, we obtained remarkably similar throughput figures in all three experiments. The rationale is that depending on many variables like network congestion, packet loss, and latency, throughput could have any value. This means that it could also be up to the value of the bandwidth but not higher. Also, here I can discuss about my results for Latency: At first, I had to run a ping test, which essentially measures RTT (Round trip time), or the amount of time it takes for data to transfer between two points and nodes. The results of this test will clearly show us how quickly packets need to be transmitted back and forth. The anticipated time between r1-2 is 10ms for each transit to and from. The process took 22.6 milliseconds (ms) instead of the predicted 20 (10ms*2). Since there are no obstructions in the way of this relatively short route, the link should just take 20ms, hence no propagation delay is anticipated. We occasionally saw a slightly larger ping than expected (up to 60ms) while the test was running and sending 25 packets between r1-2. In a scenario like this, there are various elements that could affect the connection and result in poorer performance. The

network, my device, or other elements contributing to the transmission delay could be the "problem". The same arguments also hold true for L2 and L3.

4.5 Test case 3: path Latency and throughput

4.5.1 Results

Latency	Expected RTT	RTT in reality
h1-h4	60ms	65.2ms
h1-h9	80ms	89.1ms
h7-h9	60ms	67.4ms

Throughput	Bandwidth	Throughput I reality
h1-h4	30Mbps	28.15 Mbps
h1-h9	20Mbps	16.05 Mbps
h7-h9	20Mbps	18.27 Mbps

4.5.2 Discussion

The latency in this instance was distinct from the performance testing of the link. In essence, the path delay passes via a few routers and nodes. Therefore, we should anticipate a larger RTT.
Since the transfer takes longer to reach its destination, using alternative routers would always produce worse results. In my measurement, I can observe that the predicted RTT for link connections is lower than the expected RTT for the intervals h1-h4, h7-h9, and h1-h9. I had to pass through 3 routers, L1+2 in total, which will take 30ms in each direction, to get to h4 from h1. The RTT we received is not horrible and appears to be quite standard; the additional 9ms is likely the result of network congestion or some other queue delay impact. When I measured the throughput between h1 and h9, on the other hand, the result was normal but a little higher than the predicted values. The fact that we used four routers (three connections, L1 + L2 + L3) to get to h9 added to the propagation latency, which may be an influencing factor. Here in throughput, when assessing transmission efficiency, we may also detect a decline in throughput. This indicates that the allotted time (25 seconds) is insufficient to convey as much data as I can send between two links. Here, the throughput is lower because the destination is a little further away.

4.6 Test case 4: effects of multiplexing and latency

4.6.1 Results

	Average RTT	Expected RTT	Throughput	Expected Throughput
h1-h4	73.9ms	80ms	17.29Mbps	15Mbps
h2-h5	72.8ms	80ms	11.98Mbps	15Mbps
h1-h4	70.2ms	100ms	12.47Mbps	10Mbps
h2-h5	71.5ms	100ms	14.09Mbps	10Mbps
h3-h6	72.2ms	100ms	5.46Mbps	10Mbps
h1-h4	73.5ms	80ms	18.17Mbps	15Mbps
h7-h9	70.5ms	80ms	10.95Mbps	10Mbps
h1-h4	72.4ms	60ms	28.21Mbps	30Mbps
h8-h9	25.3ms	20ms	19.08Mbps	20Mbps

4.6.2 Discussion

Here I will explain and discuss this table step by step.
When two operations are run simultaneously, there is a noticeable effect on the network performance. The Round-Trip Time (RTT) increases, and the available bandwidth for each operation is divided by 2. This means that the expected throughput of each operation is reduced. The two operations share the available bandwidth through two network layers: L1 and L2. As a result, the bandwidth is distributed between the two operations, which can lead to slower data transfer rates and longer processing times.

When three operations are run concurrently, the network performance is impacted differently compared to when two operations are run simultaneously. In this case, the available bandwidth is divided by 3, which means that the expected throughput of each operation is lower than when only two operations are run. Like the previous case, the operations share the bandwidth through two network layers: L1 and L2. This sharing can lead to a longer processing time and reduced data transfer rates.

When two operations are run on the network, the traffic is routed through different network layers depending on the source and destination of the data. Specifically, h1-h4 use the L1 and L2 layers, while h7-h9 uses the L2 and L3 layers. As a result, the two operations share only the L2 layer of the network, while the other layers are dedicated to their respective traffic. This can result in more efficient use of network resources and faster data transfer rates.

When two operations are run on the network, the traffic is routed through different paths depending on the source and destination of the data. Specifically, h1-h4 and h8-h9 have separate paths, with h1-h4 using the L1 and L2 layers, and h8-h9 using a different path that is not shared with h1-h4. This means that the two operations do not share any network resources, and each operation can utilize the full available bandwidth of its path. As a result, the data transfer rates can be faster and more efficient.

4.7 Test case 5: effects of parallel connections

4.7.1 Results

	Throughput	Expected Throughput
h1-h4	6.24 7.21	5Mbps 5Mbps
h2-h5	8.62Mbps	10Mbps
h3-6	7.56Mbps	10Mbps

4.7.2 Discussion

In this case, I repeated the same action as the previous task, with the exception that we executed one parallel connection involving three parties. The throughput yielded some intriguing outcomes. Specifically, when we gauged the simple connection between h1 and h4, we attained a throughput of nearly 6Mbps. However, when we ran the parallel connection, the resulting throughput was nearly half that amount (6.24 Mbps and 7.21 Mbps). In any case, the other tests will also have low values because of the high traffic. Based on my tests and measurements, I can conclude that the parallel connection had a minimal impact on the other connections, although it was still more significant than the impact on the parallel connection itself.

5 Conclusions

The task of measuring the ping or the Round-Trip-Time (RTT) and the throughput is an incredibly crucial aspect that requires extensive knowledge. This methodology is an effective way to analyze the network and performance evolution. In our case, we have a topology that displays the links and maximum values of the bandwidth and RTT. However, it fails to provide a clear understanding of how the connection works in different situations we must test. Therefore, conducting this procedure and being more mindful of the network connection is vital and valuable when working in this field.

While doing this project, I encountered also some problems. Two of the problems that were almost the hardest were writing the code (Simpleperf) and running the Mininet, for instance, was not an easy task. The topology code initially failed to run on my device. But in end, I had to modify the file and save it in a different location. Fortunately, the Mininet, perf, and virtual machine components proved to be very practical and useful, and they will undoubtedly aid me in future projects.

In conclusion, I have come to understand that each connection has a distinct set of conditions. When we run a connection in a User Datagram Protocol (UDP) mode, for example, it has a different functional system than Transmission Control Protocol (TCP) mode. Additionally, running multiple connections simultaneously is not the same as measuring a simple connection. Therefore, being aware of these distinctions and having the necessary tools and knowledge to tackle these issues is of utmost importance.

6 References

Contributor Staff, *Network Latency vs. Throughput vs. Bandwidth,* January 5, 2021
https://www.dnsstuff.com/latency-throughput-bandwidth

Novotný Adam, *What is RTT (Round-Trip Time) and How to Reduce it?* October 13, 2022
https://www.stormit.cloud/blog/what-is-round-trip-time-rtt-meaning-calculation/