APRIL 14, 2023

IMPLEMENTASION AND EVALUTION OF SIMPLEPERF

SIMPLEPERF A TOOL FOR MEASURING NETWORK THROUGHPUT

MISKI MOHAMAD YUSUF
S364593
Oslo Metropolitan University

1 Introduction

This study has implemented a networking performance evaluation called simpleperf, and this study aims to explore the tool. Simpleperf is simplified version of IPerf. IPerf is a commonly used network testing tool that measure the throughput of a network. It has become importance to optimizing network due to the increasing demand for high-speed network connections. The high demand means that you need more tools to check the network, which simpleperf is another tool designed to help for the increasing demand.

Simpleperf has a client and server functionality, where the client send data to server and then measures bandwidth. The user can set various parameters that can used to test network. It can choose amount of data send, how long data should be sent, number of bytes send etc.

This report explores how simpleperf is working and the functions it has. The report stat with how simpleperf was implemented, experimental setup and then evaluation of performance. It contains five test cases where various situation gets tested. Finally, will conclude with summary of findings.

2 Simpleperf

To implement simpleperf, I have created server and a client that communicate with each other. The communication is established via TCP socket, where the server is listening for incoming connections on specific port and IP port number.

The code consisted of serval functions which are divided into different parts. I implemented various options available using argpase and added them to server and the client. The options had default options, so that client and server could be invoked whit just -c and -s. In addition, I add some if statements to ensure that there is only an option to connect to either the client or the server. I also had several check functions where it checks if the port and Ip number is correct and if the input value from user is positive. I had all these checks so that the user does not write something wrong, that will make the calculation wrong.

The function of simpleperf is that user can enable the server (-s) to listen to incoming connections form clients. Client can connect to server by invoking -c. The tool allows the user to specify the total duration for how many bytes to send and total duration for data to send to sever. User can also choose the format of the summary of the result in either B, KB or MB. Simpleperf provide an interval option (-i interval flag) that allows the user to print static per z second. This can be useful for monitoring the performance of the tool over time.

The server code includes two main functions, server () and handle_client(). When server is invoked it will first check the port and IP, in the check_port() function. Then it will go to the server() functions, and it will create a socket and the socket will bind host and port. Once the socket binds Ip and port, a message will be printed indicating that the simpleperf server is listing on a certain port number. The server will then wait for new clients to connect. When client is connected, then a new thread is created to handle the client's request. The hande_client() functions is responsible for handling the request for the client and it have a while loop that receive data from the client using the recv() method. Once the client s connected to server IP address and port and the connection is established. A message is printed, and new thread is created. In the if setting where client is invoked I have I for loop where if parallel(-p) is invoked it creates a specified number of client thread that connect to a server. This allows the program to create multiple connection to the server simultaneously, with each connection running on its own thread. After the connection a new thread is created to handle the client communication. This function is client_send, where the client send data to the server. In the client_send() function it check for which options is invoked. It has several if setting where it checks for how to send data (how many bytes to send, how long to send for, if it is intervals or not). When user invoke -n (-num), then it will check which unit the user has chosen. I have made a function called formater_num(), this function will take input value and change the value to byte, this is for making calculation easier. If the user invoke -t(time), then it will send data until the time is. If -i(-interval) is invoked, it will print static per z second. I have made sure that you cannot invoke num and time at the same time. When the data is sent, a "BYE" message will be sent to the server form the client, and when the server received the message it will stop the time and send "ACK:BYE" to the client. After a result will be printed form the function print_resut() to server and client. In the print_result() function I have used tabulate for printing the result. In the function I check for if it is a client or a server and then use different headers for printing the result. Then it will print out a result for how long it went for, how many data that was sent, the IP address and port and the bandwidth/rate.

Throughout the implementation of simpleperf, I face challenges with the send function when the client transmitted data to server. I noticed that I had problem with different commands the client had available. I also had challenges with the "BYE" message sent form the client and to server. This got resolved when I avoided waiting for the "ACK: BYE" from the server and printing out the result after sending "BYE" to server. However, the time in the result showed discrepancies between the server and the client, as the connections established by the client terminated almost one second before on the server side.

3 Experimental setup

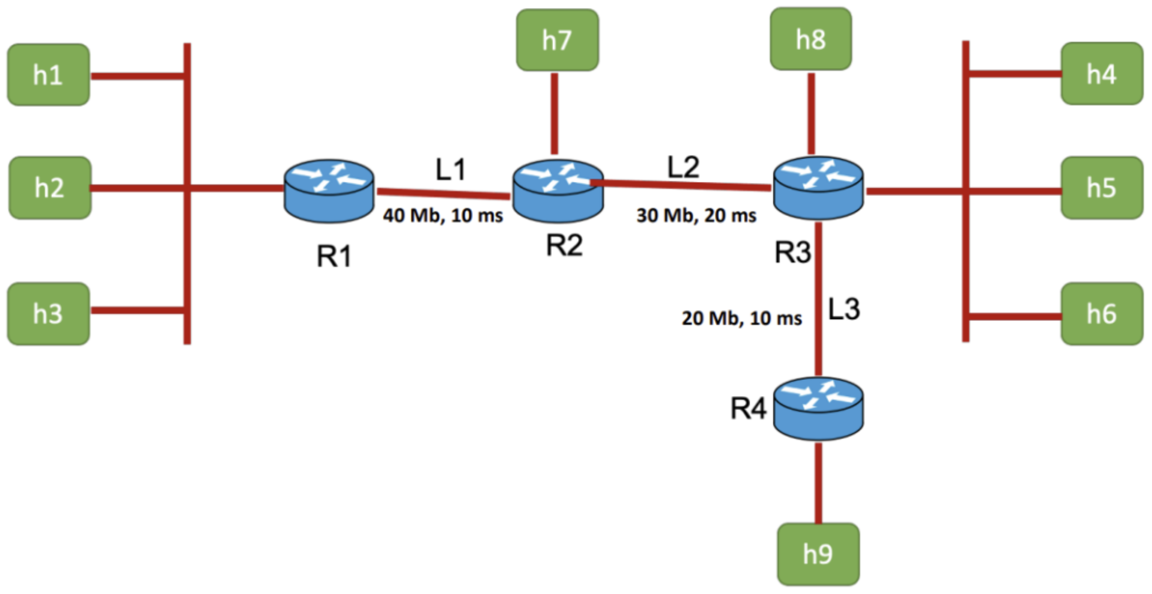

Figure 1: Network topology

Figure1 illustrates the network topology used. The network topology has eight subnets that are interconnected by four router and two switches. Subnet A consist of three hosts, h1, h2 and h3, which are linked with router R1. Subnet B connects R1 and R2, with the bandwidth of the link between them set at 40Mbps, a delay of 10ms and maximum queue size of 67 packets. Subnets C connect R2 and h7 and subnets E connect R3 and h7. Subnets D connect R2 and R3 and the link between them has bandwidth of 30Mbps, a delay of 20ms and maximum queue size of 100 packets. Subnets F connects h4, h5, h6, which are linked with router R3. Subnet G connect R3 and R4, and the bandwidth of the link is 20 Mbps, a delay of 10ms, and maximum queue size of 33 packet. Lastly, subnet H consist of R4 and h9.

4 Performance evaluations

4.1 Network tools

The network tool that been used in this experiment is iPerf, ping, simpleperf, VMware workstation 17, Mininet, xterm and visual studio code. IPerf was used to measure the bandwidth and ping was used to measure network latency and packet loss between nodes. Simpleperf was used to analyze and measure the bandwidth of network.

I attempted to use VirtualBox as a virtualization tool for my network experiments but found that the data it was providing was incorrect. Then I switched to using VMware Workstation 17, which proved to be a more reliable and accurate tool for my test. Mininet was used to simulate and test the topology. I used xterm to open terminals for each node in the Mininet network. Additionally, I used Visual Studio Code as my primary code editor for writing and editing code. The tools played a critical role in the experiments, enabling the gathering of valuable data and insights into network performances.

4.2 Performance metrics

Simpleperf is a tool to test network-related performance metrics. The following performance metrics can be measured using Simpleperf: Round-Trip-Time (RTT), Simpleperf can measure the Round-Trip Time. If the network has high RTT, it means the communication between the client and the server will have a longer delay, leading to slower response times and it will reduce overall performance. Bandwidth, simpleperf can measure the Bandwidth, which is the maximum amount of data than can be transmitted over a network in a given period of time. If the network bandwidth Is low, the application may experience slower download times or increased latency leading to reduced performance.

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

Test-case 1 in involved running three separate iPerf test with UDP mode with three different client-server pairs, with the following pair: h1-h4, h1-h9 and h7-h9. The test was performed with different MB data sent. The purpose of the test was to measure the throughput of the pairs of hosts.

4.3.1 Results

	Bandwidth	Jitter	Lost/ Total Datagram
H1-h4	28.2 Mbits/sec	0.377 ms	0%
H1 -h9	18.9 Mbits/sec	0.549 ms	0%
H7-h9	18.9 Mbits/sec	0.274 ms	0%

Table 1: throughput and total lost for test-case 1

4.3.2 Discussion

The result for test case 2 (table 1) are very like the expected result. For test case 2, I used the topology we got assigned to decide which rate of data I should measure the bandwidth. As you can see in the table the result of data in h1-h4 total loss was 0% and the bandwidth was 28.2. h1-h9 total loss was 0% and bandwidth was 18.9 Mbps. At last, h7-h9, the loss was 0% and the bandwidth was 18.9Mbps.

Questions:

1.

Bandwidth is the maximum amount of data than can be transmitted over a network in a given period of time. The topology shows us there are three links with different bandwidth. To choose which rate to measure the bandwidth you most look at the path, because the bandwidth of the path slowest link determines the maximum rate that may be measured. Example form h1 to h5 the slowest link in the path is 30Mbps, then you should select 30Mbps, but from h7 to h9 the slowest is 20Mbps so you should select 20M. However, in this test case I observed that selecting exact rate as specified in the topology resulted in packet loos. I was able to obtain 0% packet loss by lowering the rate from 30Mbps to 27Mbps. Thus, I concluded that it is preferable to select a slightly lower bandwidth rate than the specified value in the topology when calculating the bandwidth. This is something i could see in this test case, the bandwidth rate of 27Mbps was selected for h1-h4, while 18Mbps was selected for h1-h9 and h7-h9. This approach resulted in a more stable network with minimal packet loss.

2.

I am asked to use IPerf in UDP mode to measure the bandwidth in a network where I do not know anything about the network topology. I will use an exponential approach that starts with lower bandwidth and then gradually increasing the bandwidth When doing the test, I will start with low bandwidth and then look at packet loss, jitter, and delay. This will help me see which bandwidth to use. If the rate of transfer is not up to the expected level, then my next step will be to increase the bandwidth to optimize network performance. This approach is not the most effective approach, but it is better than linear testing where you start with the lowest bandwidth and then increase linear. I think this approach is the most suitable approach to take if you do not have any information about the network topology available because it allows the program to determine the optimal bandwidth to use without wasting time on bandwidth values that is too high or too high.

4.4 Test case 2: link latency and throughput

Test-case 2 was executed to measure the Round-Trip Time (RTT) and bandwidth of each of the three individual links between routers. First, I looked at L1, L2 and then L3 .The first step was to run ping with 25 packets. Then ran Simpleperf for 25 seconds for each link.

4.4.1 Results

	Average RTT(ms)	Bandwidth (Mbps)
L1	20.602	34.93

L2	40.566	28.39
L3	20.648	19.19

Table 2: The result of test case 2 when running simpleperf and ping.

4.4.2 Discussion

In the results, it can be observed that both bandwidth and the amount of data being sent are reduced with each test. This is expected given the topology assigned to us. Based on the topology, we can see that L1 has a bandwidth of 40Mbps, while L2 has 30Mbps and L3 has 20. Bandwidth available affects how much data is sent. This is also evident in my measurements where L1 had a bandwidth of 34.93 Mbps, L2 is measured 28Mbps and L3 had the lowest bandwidth of 19.19 Mbps,

In RTT, packets are measured based on how long it takes to travel from the source to the destination. In the topology L1 has delay by 10ms, L2 by 20ms, and L3 by 10ms. As we can see from the results, L1 was 20 ms, L2 was 40 ms, and L3 was 21 ms, this is a very similar result to what was expected.

4.5 Test case 3: path Latency and throughput

Test case 3 is the same as test case 2 but here I run between hosts instead of routers. In this test case I measuried RTT and throughput for three different host pairs: h1-h4, h7-h9 and h1-h9. I started by running ping server from client with 25 packets. Then, I ran simpleperf for 25 seconds for each client server pair.

4.5.1 Results

	Average RTT (ms)	Bandwith (Mbps)
H1-h4	61.339	27.01
H1-h9	82.033	16.23
H7-h9	61.320	17.01

Table 3: throughput and Round- trip time for test case 3

4.5.2 Discussion

In test-case 3 I calculate RTT and bandwidth for each pair of hosts. There are some differences between the expected and actual results. Between h1-h1 I expected a bandwidth of 30Mbps and RTT of 60ms, and the result I got was 27Mbps and 61ms, In the second test between h1-h9 I expected a bandwidth of 20 Mbps and RTT of 80ms, which is very close with the result I got. In the third test I expected 20Mbps and RTT of 60ms and I got 17Mbps and 61ms.

The results I got from the test case are similar to the expected values, however, the result did not entirely match the expected results. This could be because of hardware limitations.

4.6 Test case 4: effects of multiplexing and latency

In this experiment, I will look at the effects of multiplexing where many hosts will simultaneously communicate. Test-case 4 focuses on the effects of multiplexing on latency and throughput. It involves running different pairs of hosts simultaneously while measuring their latency and throughput using ping and simpleperf. In the first part of the test-case two pair of hosts (h1-h4 and h2-h5) will communicate simultaneously and their latency and throughput will med measured. The same test will be done for h1-h4, h2-h5, and h3-h6, as well as the hosts h1-h4 and h7-h9 and h1-h4 and h8-h9.

4.6.1 Results

Experiment 1

	Average RTT (ms)	Bandwith (Mbps)
H1-h4	71.770	16.88
H2-h5	72.033	10.34

Table 4:throughput and Round-trip time (RTT) for the pair h1-h4 and h2-h5 in experiment 1

Experiment 2

	Average RTT (ms)	Bandwith (Mbps)
H1-h4	73.580	11.73
H2-h5	72.779	7.54
H3-h6	73.453	9.19

Table 5:througput and Round-trip time (RTT) for the pair h1-h4, h2-h5 and h3-h6 in experiment 2

Experiment 3

	Average RTT (ms)	Bandwith (Mbps)
H1-h4	73.984	16.92
H7-h9	73.706	11.03

Table 6:througput and Round-trip time (RTT) for the pair h1-h4 and h7-h9 in experiment 3

Experiment 4

	Average RTT (ms)	Bandwith (Mbps)
H1-h4	81.487	27.30
H8-h9	25.007	19.11

Table 7:througput and Round-trip time (RTT) for the pair h1-h4 and h8-h9 in experiment 4

4.6.2 Discussion

When multiple pairs of hosts communicate simultaneously, they compete for the available network bandwidth. This is because they send data over the same network router and link, and both have a limited capacity of data transmission. Another factor that affects the data transmission is distance between the host and network components they are connected such as switches or routers. The maximum queue size at each device determines how much data packet that can be held in the queue before it is processed and forwarded. If the queue size is to small data may have tot wait in the queue for long time, which can increase the Round-trip time (RTT(Kurose & Ross, 2022)). This can happen if the sender send data faster than receiver can process or if there is congestion in the network.

In this test case the impact of simultaneously communication on network performance was investigated through series of experiments. The first test was between pairs h1-h4 and h2-h5. The result shows that the bandwidth decreased and the RTT increased compared to when only one pair of hosts was running. This is because when it is two pairs, they share the available bandwidth which was 30Mbps. This is also something we could see in experiment 2 where it was simultaneous communication between three pairs, where three pairs had to share 30Mbps resulting in reduced bandwidth availability.

The Round-trip time results from experiments 1 and 2 were slightly lower than what I expected. The assigned topology indicates that the queue sizes for L1, L2, and L3 are 64 packets, 100 packets, and 33 packets. Using these numbers, I can see that queue size affects the RTT of experiments 1 and 2 because multiple hosts are attempting to send data at the same time, while competing for the same network resources and having the same path between them. When there is a fixed queue size, which determines the maximum number of packet that that can be stored in the devices buffer memory at given time. A consequence is that packet will be dropped when arriving to already full buffer. But this is a reliable connection and what will happened is that the sender will eventually retransmit it (Kurose & Ross, 2022). Data is transported from L1 to L2, where L1 has 40Mbps of capacity and L2 has only 30Mbps. Therefore, the data is sent at a faster rate than the receiving router can handle, causing buffering. If the receiving router buffer becomes full, packets will be dropped, and the receiving router will stop acknowledging received packets. The sender will also stop transmitting new packets until it receives the expected acknowledgments, and it will then retransmit any lost packets. Because the sender must wait for the retransmitted packets to be acknowledged before sending new packets, this retransmission procedure might increase the RTT.

In experiment 3 the path was different in the pairs. H1-h4 and h7-h9. Here we could see that in the result it shared bandwidth for 30Mbps.The traffic passed through the same link which led to sharing of the bandwidth. Which led to the expectation of an increase in RTT due to buffering caused by sending data faster than it can be processed. However, the RTT results did not align with this prediction.

In Experiment 4, I tested the connection between h1-h4 and h8-h9. This experiment was different from, because the pairs were located at different distances from each other, and the traffic passed through different nodes. H1-h4 used connections R1-R2-R3, while h8-h9 used R3-R4, and their traffic did not cross each other because they do not share a link. Thus, I could consider these two connections as separate networks that did not compete for bandwidth and shared the available bandwidth on their respective networks. However, despite the absence of competition for bandwidth, the simultaneous transmission of these connections interfered with each other, causing an increase in the RTT due to queuing and buffering delays.

In summary, the results of the bandwidth in this test case aligned with initial expectations, whereas the RTT was not as anticipated.

4.7 Test case 5: effects of parallel connections

Test case 5 examines the effects of parallel connections on throughput. Three hosts, h1, h2, and h3, connected to R1, will communicate with three other hosts, h4, h5, and h6, connected to R3. h1 will open two parallel connections to h4, while h2 and h3 will use a single connection each. The experiment will measure the throughput when these six hosts communicate simultaneously using simpleperf.

4.7.1 Results

	Bandwidth Mbps	Bandwith Mbps (Parallel 2)
H1-h4	6.71	6.16
H2-h5	7.01	
H3-h6	9.26	

Table 8:througput for the pair h1-h4, h3-h6 and h2-h5

4.7.2 Discussion

Test case 5 was conducted to investigate the impact of parallel connections on network throughput. The experiment involved three hosts, h1, h2, and h3, connected to R1, communicating with three other hosts, h4, h5, and h6, connected to R3. Of the three hosts on R1, h1 utilized two parallel connections to communicate with h4, while h2 and h3 used a single connection each. The test gave bandwidth results of 6.71Mbps and 6.16Mbps for h1-h4, 7.01Mbps for h2-h5, and 9.26Mbps for h3-h6.

The experiment highlights the performance disparity between single and parallel connections. Parallel connections, facilitated through the use of multiple TCP sockets, increase the aggregate throughput by better utilizing the available bandwidth (Hacker et al., 2002). The results of the experiment demonstrated this assertion, with the parallel connections of h1-h4 outperforming the single connections of h2-h5 and h3-h6. While parallel connections improve network throughput, it is worth noting that they can potentially take bandwidth away from competing TCP flows when there is a limited network capacity.

5 Conclusions

In conclusion, this project explored the functionality of Simpleperf by conducting different test cases. Five test cases were tested to evaluate Simpleperf and look at the network's performance. Here I have found important result.

In Test Case 1, I observed that increasing the packet size led to a decrease in throughput, which shows me the importance of choosing the packet size to optimize network performance. I Also found out that selecting the exact bandwidth rate as specified in the topology could result in packet loss. Test Case 4 demonstrated the impact of network queue size on network performance, and it was observed that more hosts connected to a network could increase round-trip time and decrease bandwidth. These findings emphasize the importance of optimizing the queue size to minimize the round-trip time and ensure efficient data transmission in a network. Additionally, Test Case 5 demonstrated the effectiveness of parallel connections in improving network throughput. There are many factors that can affect network performance, and Simpleperf only measures bandwidth without taking into consideration network congestion and packet loss. Despite these limitations, Simpleperf proved to be a useful tool for evaluating network performance.

6 References (Optional)

Hacker, T. J., Athey, B. D., & Noble, B. (2002). The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. *Proceedings 16th International Parallel and Distributed Processing Symposium*, 10 pp. https://doi.org/10.1109/IPDPS.2002.1015527

Kurose, J. F., & Ross, K. W. (2022). *Computer networking: A top-down approach* (Eighth edition. Global edition). Pearson Education Limited.