APRIL 14, 2023

IMPLEMENTASION AND EVALUTION OF SIMPLEPERF

[DOCUMENT SUBTITLE]

MISKI YUSUF MOHAMAD
S364593
Oslo Metropolitan University

1 Introduction

The goal of my project is to implement Simpleperf, a simplified version of iPerf, which is a network performance evaluation tool that measures network latency and throughput. The importance of optimizing network performance has become increasingly crucial due to the growing demand for high-speed network connections. This study aims to explore the network performance evaluation tool Simpleperf.

Simpleperf functions by creating a server-client mode, where the client sends data to the server and measures the data sent. The user can specify several parameters, such as the amount of data, the number of bytes, and the duration of data transfer. Simpleperf has certain limitations and outcomes, as it only measures network bandwidth and does not take into account other factors, such as network congestion, packet loss, and latency, which can affect network performance.

This report explores the features of Simpleperf and demonstrates how it can be utilized to analyze network performance. It includes a description of the implementation of Simpleperf, the experimental setup, and an evaluation of performance. The report concludes with five test cases that assess Simpleperf performance in various situations. Finally, a summary of findings will be presented.

2 Simpleperf

To implement Simpleperf, I first created a server and a client that communicate with each other. The code consists of two parts, the server part and the client part. The communication is established via TCP socket, where the server listens for incoming connections on a specific IP address and port number. The server code includes two main functions: server () and handle_client(). When a server is invoked, it will first check the port and Ip address in the function check port() and Ip address. Then it will go to the function named server and it will create a socket, and the socket will bind host and port. Once this is completed, a message is printed stating that the Simpleperf server is listening on a specific port number. The server waits for new clients to connect, and when a client is connected, a new thread is created to handle the client's request.

The handle_client() function is responsible for handling the request from the client, and the server accepts the connection using the accept() method and receives data from the client using the recv() method. Once the client is connected to the server's IP address and port number, and once the connection is established, the client sends data to the server. In the client_send() function, it checks for which sending function to call and how to send data, such as how many bytes to send, how long to send for, and intervals. If the user invokes the -num option, then the formater_num function is called to convert the value to bytes. If the user invokes the -time option, the client sends data until the specified time has elapsed. If the user invokes both -time and -interval, data is sent, and the result is printed at the specified interval. However, invoking both -num and -time in the client_send() function is not allowed.. After the data is sent, the client sends a "BYE" message to the server, and when the server receives this message, it stops the timer and sends an "ACK:BYE" message to the client. The print_result() function prints out the result on both the server and client sides.

I implemented various options available for the server using argparse and added them to the server and the client. Furthermore, I introduced conditional statements to ensure that exclusively one option is selected at a time, either connecting as a client or operating as a server. The primary purpose of the Simpleperf tool is to allow users to establish server mode (-s) to receive incoming connections from clients, who in turn, connect to the server by invoking -c. This tool also empowers the user to determine the duration for which data should be transmitted to the server. Additionally, the format of the summary of results can be chosen as B, KB, or MB. Simpleperf further incorporates an interval option (-i interval flag), enabling users to obtain statistics periodically, which could facilitate performance monitoring over time.

During the implementation process, I encountered several challenges while employing the send function, particularly when the client transmitted data to the server. Specifically, the client's diverse available commands created performance issues, which resulted in unsatisfactory testing outcomes. I also encountered difficulties with the Bye message sent from the client to the server, which were only resolved when I avoided waiting for the ACK: BYE from the server and printed out the result after receiving the BYE from the client. However, the time results showed discrepancies between the server and the client, as the connections established by the client terminated almost one second before those on the server side.

3 Experimental setup

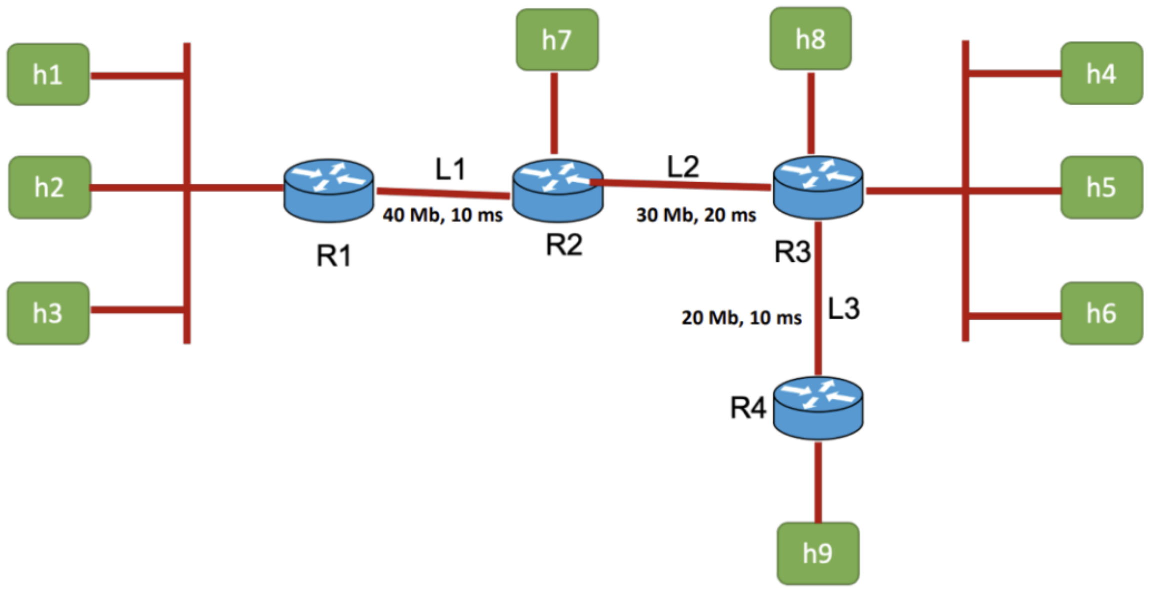

Figure 1: Network topology

Figure 1 illustrates the network topology assigned to assess the functionality of the simpleperf tool. This topology was selected to enable the evaluation of simpleperf in a network with multiple hosts and routers. The network topology encompasses eight subnets that are interconnected by four routers and two switches. Specifically, subnet A consists 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 a maximum queue size of 67 packets. Additionally, subnets C and E connect R2 and h7, and R3 and h8, respectively. Subnet D connects R2 and R3, and the link between them is characterized by a bandwidth of 30Mbps, a delay of 20ms, and a maximum queue size of 100 packets. Subnet F comprises h4, h5, and h6, which are linked with router R3. Subnet G connects R3 and R4, and the bandwidth of the link between them is set at 20Mbps, a delay of 10ms, and a maximum queue size of 33 packets. Lastly, subnet H consists of R4 and h9.

4 Performance evaluations

4.1 Network tools

A variety of network tool have been used in this experiment to gather data and analyze network performance. The tools I have used is ping, iPerf, simpleperf, VMware workstation 17, Mininet xterm, visual studio code. Ping was used to measure network latency and packet loss between nodes. iPerf was used to measure the bandwidth and Simpleperf was used to measure and analyze the performance of the program, including network related performance.

I attempted to use VirtualBox as a virtualization tool for my network experiments but found that the data it was providing was incorrect. As a result, I switched to using VMware Workstation 17, which proved to be a more reliable and accurate tool for my experiments. Mininet network emulator 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 related to the network tools and protocols we were testing.

The tools played a critical role in the experiments, enabling the gathering of valuable data and insights into network performance. The combined utilization of these tools facilitated a comprehensive analysis of the network, allowing for the identification of areas in which improvements could be implemented to optimize network performance.

4.2 Performance metrics

Simpleperf is a tool to evaluate network-related performance by measuring two performance metrics: Round-Trip Time (RTT) and Bandwidth. RTT is a measure of the time it takes for a client to send a request to a server and receive a response back. This metric can indicate the delay in communication between the two devices. Higher RTT indicates longer response times, slower download speeds, and ultimately reduced network performance. On the other hand, bandwidth refers to the maximum amount of data that can be transmitted over a network in a given period of time. If the network bandwidth is low, this can result in slower download times, increased latency, and ultimately reduced overall 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 transmitted over a network per unit of time. In the given topology, there are three links with different bandwidths, which implies that the bandwidth is limited by the slowest link in the path. Therefore, the maximum rate that can be measured is equal to the bandwidth of the slowest link in the path. To choose this you must look at the path, for example from h1 to h4, if the slowest link in the path is 30, then you have to select 30Mbps. But from h7-h9 the lowest is 20 Mbps, so you have to select that. In this particular test case, I observed that selecting the exact bandwidth rate as specified in the topology resulted in a considerable amount of packet loss. However, by slightly reducing the rate from 30Mbps to 27Mbps, I was able to achieve a 0% packet loss. 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. Specifically, 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.
If I am asked to use iPerf in UDP mode to measure bandwidth in a network where I do not know anything about the network topology, I will use an exponential approach that starts with a lower bandwidth rate and then double increase till I reach maximum available bandwidth. When doing the test, I will look at packet loss, jitter and delay, because that will help me see which bandwidth to use. I will conduct many tests while examining different bandwidths and then monitor network conditions. That will help me pick and choose the best bandwidth for the network. 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. This is because it is time-consuming. I think this approach is the most suitable approach to take if you do not have any information about the network topology available.

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 between R1 and R2, then L2 between R2 and R3, and finally L3 between R3 and R4. 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	Bandwith 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

The results reveal a reduction in both bandwidth and the amount of data transmitted with each test, which is in line with the assigned topology. The topology features L1, L2, and L3 with bandwidths of 40Mbps, 30Mbps, and 20Mbps, respectively. The amount of data sent is affected by the available bandwidth, which is also evident in the measurements obtained, where L1 had a bandwidth of 34.93Mbps, while L3 had the lowest bandwidth of 19.19Mbps. As expected, the direct connection between L1 and L2 has the highest bandwidth, whereas the connection between L2 and L3, which is connected through R3, has lower bandwidth. The measured bandwidth for L1, L2, and L3 was closer to the set values of 40Mbps, 30Mbps, and 20Mbps.

Regarding the Round-Trip Time (RTT), it measures the time taken for a packet to travel from source to destination and back again. The assigned topology has L1, L2, and L3 with delay values of 10ms, 20ms, and 10ms. The measured RTT values of 20ms, 40ms, and 20ms for L1, L2, and L3 respectively, align with the expected results.

While the observed results are similar to the expected outcomes, there are little differences. They may be several factors that contribute to the differences between the expected result and the actual results. One such factor is the limitations of hardware used in the test case.

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. It involves measuring the path latency and throughput for three different host pairs: h1-h4, h7-h9 and h1-h9. Ping and simpleperf got used to measure the latency and throughput. 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	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

4.5.2 Discussion

In test-case 3 Round Trip Time (RTT) and bandwidth were calculated for each pair of hosts. While there were similarities between the results and the expected values, there were also some differences between them. For the first test between h1-h1 I expected a bandwidth of 30Mbps and RTT of 60 ms. And the result I got was 27Mbps and 61ms. However, the measured values were 27Mbps and 61ms, respectively, indicating a slight deviation from the expected outcome.

Similarly, the second test between h1-h9 had a bandwidth of 16.23 Mbps and an RTT of 82ms, which were very close to the anticipated values. In the third test, an expected bandwidth of 20Mbps and an RTT of 60ms were projected, but the measured values were 17Mbps and 61ms, respectively.

Although the obtained results were largely in agreement with the expected values, it is important to note that there were discrepancies between them. This discrepancy may be attributed to a variety of factors that could be contributing to slower data transfer rates and higher delays. Such deviations may be attributed to various factors that can affect network performance, including 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 process will be repeated for three pairs of hosts (h1-h4, h2-h5, and h3-h6) and for pairs of hosts h1-h4 and h7-h9 and h1-h4 and h8-h9.

4.6.1 Results

Experiment 1

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

Table 4:througput and Round Trip Time (RTT) for the pair h1-h4 and h2-h5 in experiment 1

Experiment 2

	Average RTT	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	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	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 as they send data over the same network link and router, both of which have limited capacity for data transmission. As a result, the bandwidth available to each pair of hosts decreases, leading to slower transfer rates, longer transfer times, and poorer performance. Moreover, the distance between the hosts and the network components they are connected to, such as switches or routers, can also affect the data transmission process. Specifically, the queue size at each device can determine how much data can be held in the queue before it is processed and forwarded, and if the queue size is too small, data packets may have to wait in the queue for a longer time, which can increase the round-trip time. This can happen if the sender sends data faster than the receiver can process it, or if there is congestion in the network. As a result, the sender may have to wait for an acknowledgement from the receiver, which can increase the round-trip time.

In the present study, the impact of simultaneously communication on network performance was investigated through a series of experiments. The first experiment involved the pairs h1-h4 and h2-h5, where the bandwidth decreased, and the round-trip time increased compared to when only one pair of hosts was running. The experiment revealed that the pairs shared the available bandwidth of 30Mbps, which was also observed in the second experiment, where three pairs had to share 30Mbps resulting in reduced bandwidth availability.

The Round-Trip-Time (RTT) 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. In the topology, data is transferred from L1 to L2, where L1 has a bandwidth of 40Mbps 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. This retransmission process can increase the RTT as the sender must wait for the retransmitted packets to be acknowledged before sending new packets, thereby increasing the RTT.

In experiment 3 the path was different in the pairs. where h1-h4 and h7-h9 utilized the R1-R2-R3 connection, while h7-h9 utilized R2-R3-R4. As a consequence of sharing the same link, a bandwidth of 30Mbps was shared among them. Which led to the expectation of an increase in the round trip time (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 Experiments 1,2 and 3 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, namely h1, h2, and h3, connected to R1, communicating with three other hosts, namely 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 experiment yielded throughput 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 corroborate 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. Alternative approaches aimed at improving performance have been proposed, but all of them suffer from the same issue of increased effectiveness at the expense of fairness.

5 Conclusions

In conclusion, this project has aimed to explore the capabilities of Simpleperf by conducting a series of experiments with different test cases. Through the implementation of Simpleperf and the experimental setup, five test cases were evaluated to assess network performance in various scenarios. The findings of this project have important implications for optimizing network performance in high-speed network environments.

In Test Case 1, the experiment observed that increasing the packet size led to a decrease in throughput, which highlights the importance of carefully choosing the packet size to optimize network performance. The experiment also found that selecting the exact bandwidth rate as specified in the topology could result in packet loss. Furthermore, 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. This finding suggests that utilizing multiple parallel connections can be an effective way to enhance network performance. However, it is important to note that Simpleperf measures network bandwidth but does not account for factors such as network congestion and packet loss, which can also affect network performance.

Despite these limitations, Simpleperf proved to be a useful tool for evaluating network performance in the tested scenarios. Overall, this project has demonstrated the effectiveness of Simpleperf in assessing network performance and highlights the importance of optimizing network performance in today's high-speed network environments. The findings of this project provide valuable insights for network administrators and researchers to optimize 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