APRIL 16, 2023

NETWORK PERFORMANCE ANALYSIS

ASSESSING LATENCY, THROUGHPUT, AND THE IMPACT OF MULTIPLEXING AND PARALLEL CONNECTIONS

GAUTE ZHANG KJELSTADLI
S362066
Oslo Metropolitan University

1 Introduction and methodology

The need for high-speed networks has grown exponentially in recent years. The rapid rise of internet-enabled applications like communication channels, entertainment platforms and e-commerce sites have contributed significantly to this increase in demand.

In light of this trend, it's essential to comprehend network performance metrics such as bandwidth, latency and throughput to identify potential bottlenecks that hinder smooth operations. In this report, I will present a possible solution for accurately measuring these parameters within a controlled environment using Mininet while drawing from existing resources such as *iperf, ping* and *simpleperf*.

The methodology employs an approach comprising several different experiments applied to a pre-determined network topology involving several client-server pairs along with varied test scenarios. With these trials, we intend to assess various components within the system, primarily focusing on link and path latency, throughput, and exploring any impacts seen through multiplexing, alongside parallel connections compared to single connections regarding overall performance.
The objective for conducting these examinations is acquiring insightful knowledge about underlying factors affecting internal network behaviour.

However, it is essential to acknowledge the limitations of our study. The experiments are conducted in a controlled environment using Mininet on an Ubuntu operating system within a Virtual Machine (VirtualBox) which may not fully represent real-world network conditions. Furthermore, the results are subject to factors such as traffic and hardware capabilities.

1.1 Preliminary information and assumptions

During the initial testing attempt, an obstacle was encountered on my computer (Dell XPS 15), seemingly due to the hardware or setting on the host operating system. Consequently, Diba Shishegar's MacBook Pro (m1 processor) was utilized to conduct the tests. Subsequent to completion of the tests, it was observed that the unit of measurement for throughput was indicated in MBps instead of Mbps, despite the fact that the throughput was displayed in Mbps. Given that the tests had already been carried out on Diba's computer, it was deemed impractical to modify the unit of measurement at that juncture. Furthermore, it is worth noting that certain outcomes may have been more precise to the expected outcome if Diba's computer had been available throughout the testing process. Regrettably, this was not feasible, and thus it is possible that certain findings may be suboptimal for the purpose of this report.

Furthermore, during the testing process, it was observed that the server failed to print the *simpleperf* results when directing the client results to a text file. Additionally it was observed that it was necessary to manually interrupt the process after the specified time by pressing "CTRL-C". Notably, this was not required when the measurements were not being directed to a text file. These observations suggest that the behaviour of the *simpleperf* tool may be influenced by the method used to transmit data, and highlight the importance of careful attention to the details of experimental methodology in achieving accurate and reliable results. It should also --be noted that when utilizing the -P flag with *simpleperf*, it was observed that the table headers were printed multiple times, with one header printed for each connection. However, given that this was deemed a purely cosmetic issue, no action was taken to modify the code to print one header.

Additionally, it is important to note that potential hardware limitations may have exerted an influence on the results of this evaluation.

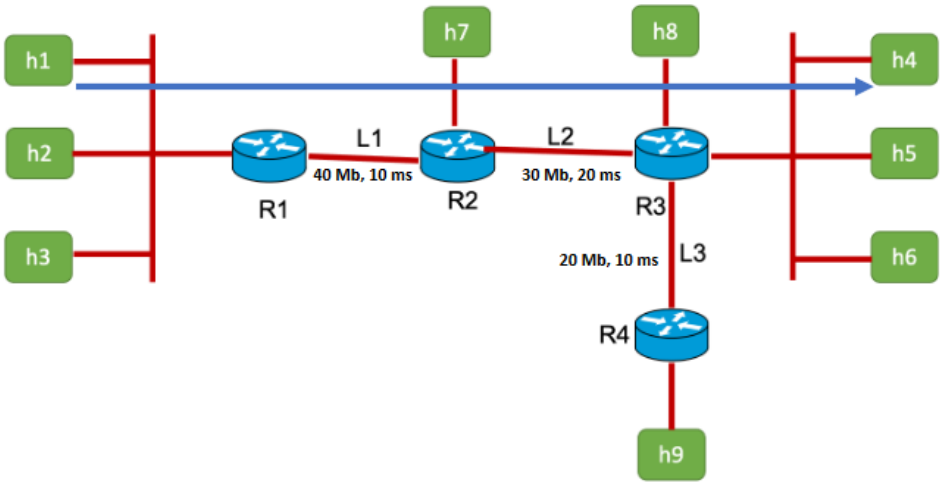
2 Simpleperf

Here's a summary of the *simpleperf* tool (The README.md file contains a more in depth description):

This implementation of *simpleperf* is a network performance tool that enables users to assess network bandwidth and throughput in both server and client modes. The tool listens on a specified port and records data received from client connections in server mode. In client mode, the tool allows multiple connections to the server and sends data over a specified time interval. The tool provides statistics on data transfer and throughput, which can be displayed in a specified format (B, KB, MB). The tool also enables users to specify number of bytes to send, and the interval at which to display the statistics. Overall, this implementation of *simpleperf* can be useful in measuring and evaluating network performance, which I will demonstrate in this report.

3 Experimental setup

The network topology that was provided for this study is a intricate network that is designed to emulate a system with multiple paths and link characteristics. The hosts are linked to the switches, which in turn were connected to the interconnected routers, forming a hierarchical structure. Throughout the testing, I applied specific configurations to emulate different network conditions and to evaluate the impact of these factors on network performance.



4 Performance evaluations

4.1 Network tools

In this study, we utilized various network tools to evaluate and examine the performance of different network configurations and topologies. One of these tools is *Ping*, a method used to determine latency, or the amount of time it takes for data to travel between two devices across a network. Low ping times are crucial when timely delivery of data is essential (Adiyatri, 2021). Another tool employed in our experiments is *IPerf*/*IPerf3*, which is used to measure the bandwidth on IP networks. It can be adjusted to test different parameters like timing, buffers, and more, and it provides a detailed report on bandwidth, loss, and other metrics after each test.

As mentioned earlier, our experiments were conducted using *Mininet*, an open-source network emulator, running on an *Ubuntu* operating system within a *Virtual Machine* (*VM*) by *VirtualBox*. The combination of these tools enabled us to effectively simulate and explore various network scenarios in a controlled and efficient manner.

Understanding the *OSI* model (Open Systems Interconnection) is crucial for our analysis, as it is a universal computer networking model that allows different communication systems to communicate using standard protocols. The model is based on splitting a communication system into seven abstract layers, with each layer having specific functionality (GeeksforGeeks, 2023). *TCP* and *UDP* are both examples of transport layer protocols in the *OSI model*. *TCP* (Transmission Control Protocol) is a connection-oriented protocol, while *UDP* (User Datagram Protocol) is a connectionless protocol that operates on top of *IP*. Unlike *TCP*, *UDP* is unreliable and connectionless. *TCP* establishes an end-to-end connection between the sender and receiver, ensuring data is delivered in the correct order and without loss. In contrast, *UDP* provides no guarantee that packets will arrive in the correct order or even at all.

The *simpleperf* implementation is based on *TCP*, which guarantees that data packets are delivered in the correct order, without duplicates or missing parts. If a packet is lost or corrupted, it will be retransmitted. This could explain why the bandwidth might not be as fast as with *UDP*, which does not guarantee reliable delivery of packets.

4.2 Performance metrics

In this section, we will explore the key indicators that were employed throughout the evaluation process. These metrics are vital for understanding the network's behaviour under various network configurations.

RTT is a term utilized in this report to describe the duration required for a data packet between two "nodes" in a network. This metric is a measure of **latency**, which represents the required duration for a packet to travel from one destination to another. Ideally, latency and RTT will be as close to zero as possible. (Gillis, A. 2020)

Throughput, is a metric meant to describe the volume of data that is successfully transmitted over a network within a given duration. In the context of this study, the throughput is represented in megabits per second (Mbps). **Bandwidth**, however, denotes the maximum data transfer capacity of a network within a given time period. (Indeed Editorial Team, 2023)

Packet loss is a term used to describe the instance when one or more transmitted data packets fail to arrive at their end destination (Gillis, A. 2021)

The **units** that were used in this evaluation report are MB and Mbps. MB (megabyte) is a measurement of the data transfer and Mbps (megabits per second) is a measurement for network speed. It is important to note that one megabyte is equal to eight megabits: 1MBps = 8Mbps.

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

In test-case 1 , we aim to evaluate the performance of the network by examining the bandwidth between different host-pairs with contrasting bottlenecks using the *Iperf* tool in *UDP* mode. This analysis was carried out with emphasis on having a bandwidth as close to the bottleneck as possible as well as having the smallest possible Lost/ Total Datagrams ratio. This ratio indicates the proportion

of lost datagrams (packets) in the network, which can have an impact on the performance and network efficiency for the application. Therefore any packet loss is generally undesirable, as it can lead to poor performance in applications and reduced network efficiency. (Lamberti, 2023).

1) **Which rate (X) would you choose to measure the bandwidth here?**

In a network, the bottleneck is determined by the link with the lowest bandwidth. In the case of the first test h1-h4, the L2 link the lowest bandwidth at 30 Mbps. Therefore, the bottleneck from h1 to h4 is 30 Mbps.
r1 to r2: 10.0.1.1/24 subnet, bandwidth is set to 40 Mbps (bw=40).
r2 to r3: 10.0.3.1/24 subnet, bandwidth is set to 30 Mbps (bw=30).
Similarly, we could determine the other bottlenecks using the same method
Given the bandwidths, we can choose a rate (X) that is similar to the bottleneck bandwidth to saturate the link and get accurate measurements. For instance:

- X = 30 for the h1-h4 pair.
- X = 20 for the h1-h9 pair.
- X = 20 for the h7-h9 pair.

For each test, I started the server on the host destination. Correspondingly I run the client on the source host. When running the test I specified the period as 25 seconds with the -t 25 argument due to getting irregular results when choosing a smaller period like -t 10. Furthermore, A minimal packet loss rate (below 1-2%) is deemed tolerable for the majority of applications (Lamberti, 2023). Therefore each test was executed with the aim of getting a packet loss ratio of below 1-2%.

4.3.1 Results and discussion

H1-H4 results
Iperf -c 10.0.5.2 -u -b 30M -t 25 > throughput_udp_iperf_h1-h4.txt

```
[ 1] 0.0000-25.0178 sec 87.0 MBytes 29.2 Mbits/sec 0.047 ms 4852/66877
(7.3%)
[ 1] 0.0000-25.0178 sec 16 datagrams received out-of-order
```

The approach I opted for, involved testing the bandwidth as closely as possible to the bottleneck. In this case the bottleneck is 30, and the test was performed by specifying the bandwidth as 30M. The results of the test showed a throughput of 29.2 Mbps, which is close to the maximum bandwidth capacity of the bottleneck link. However, the packet loss rate was 7.3%. Which is higher than the desired range of below 2%. Therefore I conducted another test with a bandwidth of 27M:

```
[  1] 0.0000-24.9979 sec  78.1 MBytes  26.2 Mbits/sec   0.047 ms 0/55731
(0%)
[  1] 0.0000-24.9979 sec  12 datagrams received out-of-order
```

The average throughput was 26.2 Mbps, which is lower than the previous test. However, the packet loss rate was reduced to 0%, which Is a significant improvement compared to the previous test. Additionally, there were still some out-of-order packets received, although the number was relatively small. Overall, the test results suggest that reducing the bandwidth to 27Mbps has improved the packet loss rate, therefore I conducted the remaining tests with by specifying a bandwidth that is slightly lower than the bottleneck link.

H1-H9 results
We already know from previous tests on h1-h4, that we should specify the bandwidth rate slightly below the bottleneck to decrease the packet-loss-rate

```
[  1] 0.0000-24.9978 sec  56.3 MBytes  18.9 Mbits/sec   0.006 ms 0/40127
(0%)
[  1] 0.0000-24.9978 sec  11 datagrams received out-of-order
```

The *iperf* test for h1-h9 was specified with the bandwidth as -b 18M, slightly below the bottleneck of 20M. The result shows an average throughput of 18.9Mbps. Additionally there were 11 datagrams received out-of-order. Overall, the results suggest that the network performance for h1-h9 is stable and the specified bandwidth rate is effective in reducing the packet loss rate.

H7-H9

```
 [  1] 0.0000-25.0005 sec  56.3 MBytes  18.9 Mbits/sec   0.001 ms 0/40127
(0%)
```

The test result of h7-h9 are similar to the result of the test on h1-h9 when specifying the bandwidth as 18M.

Based on the results of the three *iperf* tests, it is clear that specifying the bandwidth slightly below the bottleneck is an effective approach to reducing the packet loss rate. The first and the second test (h1-h4, h1-h9) resulted in a higher out-of-order datagrams. Although this number was quite low, it may be due to network path issues.

2)
When asked to use Iperf in UDP mode to measure the bandwidth when the network topology is unknown, I would start by identifying the endpoints and choose two endpoints within the network. Furthermore, I would start with a low bandwidth value and incrementally increase it in steps, running Iperf tests at each step. Simultaneously monitoring the packet loss percentage and observe at which point the packet loss starts to significantly increase. This point will give an estimation of the maximum achievable bandwidth. However, this approach might be an unpractical approach as it can generate unnecessary traffic and potentially disrupt the network. Additionally it may take a long time to converge to the correct value. Therefore it is optimal to use this approach while taking precautions to minimize the impact on the network, such as scheduling the tests during low-traffic periods, limiting the duration and the frequency of the tests.

4.4 Test case 2: link latency and throughput

In this task, the objective is to measure the round-trip time (RTT) and the bandwidth of three individual links(L1, L2, L3). To achieve this, we will run the *ping* command with 25 packets to measure the RTT. Additionally, we will run the *simpleperf* tool for 25 seconds to measure the throughput of each link. The test results will then be compared to the results of the previous test case to gain a better understanding of the network performance.

4.4.1 Results and discussion

Based on the topology, the expected *RTT* values will differ for each of the router links. These are the expected RTT values:

L1: 2*10ms = 20ms, **L2:** 2*20ms = 40ms, **L3:** 2*10ms = 20ms

The expected results for the throughput values are based on the bottlenecks provided in the topology. Based on the results from test case 1, we expect a slightly lower throughput than the bottleneck:

L1: Approx 39Mbps, **L2:** approx. 29Mbps, **L3:** approx. 19Mbps

Test results:

Link	L1	L2	L3
Latency avg.	22.67ms	44.75ms	22.83ms
Throughput	38.65Mbps	29.12Mbps	19.24Mbps

L1: The measured throughput for L1 is close to the expected bandwidth, indicating a stable and efficient network performance. The measured RTT is slightly higher than expected. This difference could be attributed to factors such as processing delays at the routers and/or network congestion.

L2: Similarly, the measured throughput is closed to the expected bandwidth, showcasing stable network performance. The RTT is comparably higher than expected, which could be attributed to the same factors as for L1.

L3: Similar results and conclusion as the results of L1 and L2.

In summary, the measured throughput values for the router links are close to the expected bandwidths, indicating a stable network performance. The measured RTT values are slightly higher than expected, which could be due to factors such as processing delays at the routers and network congestion. On the whole, these results provide valuable insight into the network performance and can be used to identify areas for optimization.

4.5 Test case 3: path Latency and throughput

Similarly to test case 2, we aim to examine network performance with emphasis on path latency and throughput, However, in this section we will measure the latency and throughput between specific host pairs and not router links. We will focus on three different scenarios, using *ping* with 25 packets and *simpleperf* for a duration of 25 seconds. Upon completion of the tests, we will discuss the results and compare them to the expected outcome to understand the network performance between the specific host pairs.

4.5.1 Results and discussion

The expected result is based on the same calculation method that was used in test case 1. In other words, the expected outcome is a result of the bottlenecks in the topology. Additionally we expect a throughput that is slightly lower than the bottleneck of the link with the lowest bandwidth (test case 1).

Expected outcome:

Host-pairs	H1-h4	H7-h9	H1-h9
Latency	(10ms+20ms)*2=60ms	(20ms+10ms)*2=60ms	(10ms+20ms+10ms)*2=80ms
Throughput approx.	29Mbps	19Mbps	19Mbps

Test results:

Host-pairs	H1-h4	H7-h9	H1-h9
Latency avg.	68.23	66.36	88.81
Throughput	28.85Mbps	19.21Mbps	19.09Mbps

Discussion

H1-h4: The expected latency was 60ms, while the measured average latency was 68.23ms. The observed latency is higher than expected, which could be due to factors such as processing delays at routers and network congestion. Furthermore, the expected throughput was approx. 29Mbps, and the measured throughput was 28.85Mpbs. The close match between expected and observed throughput indicates that the network resources are being utilized efficiently, and the bottleneck link is performing close to its maximum capacity.

H7-h9: Similarly to h1-h4, the measured latency of 66.36ms was slightly higher than the expected latency. Similar factors affecting h1-h4 could be at play here, resulting in a similar result. The expected throughput was approx. 19Mbps, and the measured throughput was slightly higher than expected at 19.21. This result, could be attributed to a more efficient utilization of the available bandwidth or less congestion.

H1-h9: In this scenario, we get similar results for both the latency and the throughput. Comparably, the higher latency suggests more constraints in the network path, such as congest network segments, slower links or additional processing delays. Additionally, the throughput result is slightly higher than the expected outcome for the same reasons as in h7-h9.

In conclusion, the test results show that the latency values are higher than the expected outcome for all host pairs, while the throughput values are generally close to the expected outcomes. These discrepancies in latency match with the results in test case 2 and they could be attributed to network path constraints, processing delays or congestion. The throughput results, on the other hand, suggests that the network resources are being utilized efficiently, similarly to test case 2. Overall, the network performance can be considered satisfactory, although it might be possible to optimize the performance further by addressing the factors causing a slightly higher latency.

4.6 Test case 4: effects of multiplexing and latency

In this section, I will investigate the effects of multiplexing on network performance by running experiments with multiple host pairs communicating simultaneously. Additionally I will measure latency by using *ping* and throughput by using *simpleperf* for various scenarios with different host pair combinations.

4.6.1 Results and discussion

Contradictory to the previous test cases, when performing the tests, the ping command were started almost simultaneously as simpleperf. As a result, the latency measurements is expected to be higher than in previous tests. Comparably, the throughput measurements is expected to be lower than previously. This is because both commands were executed at the same time, leading to increased contention for network resources and potential delays in packet transmission.

The expected throughput value for the different multiplexing scenarios is dependent on the bottleneck for the network path. Similarly to previous test cases, the bottleneck is determined by the link with the lowest bandwidth:

Scenario 1: approx. 29Mbps/2 = 14.5Mbps.
Scenario 2: approx. 29Mbps/3 = 9.67Mbps.
Scenario 3: 29Mbps/2 = 14.5Mbps.
However, since the host pairs don't share the same path, the bandwidth will likely differ from this.

Scenario 4: 29Mbps for h1-h4 and 19Mbps for h8-h9

It is important to note that the timing of these commands might not have been perfectly synchronized. This could introduce some variability in the results and make it harder to draw accurate conclusions about the effects of multiplexing on network performance. Despite these limitations, this experiment will still provide valuable insight to the behaviour of the network when multiple host pairs communicate simultaneously.

It is also **worth noticing** that hardware capabilities may have had more of an impact on these results. Additionally, the tests executed in ascending order, meaning that I started to measure for H1-h4, then h2-h5 (scenario 1). Which could have resulted in better performance for h1-h4 than h2-h5.

Results:

Scenario	1		2			3		4	
Host-pairs	H1-h4	H2-h5	H1-h4	H2-h5	H3-h6	H1-h4	H7-h9	H1-h4	H8-h9
Latency avg.	104.04ms	103.45ms	102.52ms	100.26ms	104.76ms	90.40ms	92.29ms	99.49ms	39.32ms
Throughput	17.57Mbps	11.97Mbps	9.49Mbps	7.35Mbps	12.92Mbps	18.96Mbps	10.57Mbps	29.10Mbps	19.25Mbps

Discussion:

When we compare the results the following patterns become apparent:

Scenario 1:
In this scenario, two pairs of hosts communicate simultaneously, both using the same network path. The communication path for both pairs likely experiences congestion, resulting in increased latency and decreased throughput, when compared to the previous results. The lower throughput for h2-h5 could be due to variations in network conditions or the specific time at which the communication occurred.

Scenario 2:
When three pair of hosts communicate simultaneously, the network experiences an even greater congestion, as there are more hosts sharing the same network resources. This results in higher latency and lower throughput for all host pairs compared to scenario 1, as all the pairs share a common network path that likely experiences greater congestion. The increase in latency and decrease in throughput is more pronounced for h2-h5 and h3-h6, likely due to reasons stated in the "it is worth noticing" paragraph.

Scenario 3:
In this scenario, we test for two pairs of hosts similarly to scenario 1 but the two host pairs use separate network paths for communication. The expected outcome is better performance than the scenario 1 and worse performance than previous tests because the host-pairs don't share the same

network path. The measurements show a lower latency value for both host pairs compared to the previous scenarios. This could be due to different paths or less congested routes being used for communication. Although the throughput value for h1-h4 seems higher than in scenario 1, the h7-h9 host pair experiences a lower throughput expected. This is likely a result of network congestion on the L2 link and for reasons specified in the *worth noticing* section.

Scenario 4:
In this final scenario, the latency value for h1-h4 is similar to scenario 1, while latency for h8-h9 is significantly lower. This suggests that the network paths for h8-h9 may be less congested or have a more direct route. The throughput for both host pairs is higher than in previous scenarios, in fact it is close to the throughput that we would expect when measuring throughput on individual host pairs. Using the same method as in test case 1 (bottleneck on the link with the lowest bandwidth), we would expect a throughput of approx. 29Mbps on h1-h4, and approx. 19Mbps on h8-h9. This indicates that the network resources are being utilized more efficiently when the host pairs don't share the same network path.

In **conclusion**, the results show that multiplexing affects both latency and throughput in the network. As more host pairs communicate simultaneously, network resources are shared, leading to lower throughput values. RTT values are also impacted by simultaneous communication but appear to be more stable across scenarios. Furthermore, it becomes apparent that when the host pairs share the same network path, they compete for the same network resources, leading to a negative impact on network performance. In contrast, when host pairs use separate network paths, the distribution of network resources can be more balanced, potentially leading to better performance for both pairs. These findings highlight the importance of efficient network resource management, such as load balancing and traffic shaping to ensure optimal performance in multiplexing situations Additionally, understanding the network topology and the specific paths utilized by different host pairs can provide valuable insights into potential bottleneck and areas for network optimization. Understanding the effects of multiplexing is crucial for optimizing network performance and resource utilization.

4.7 Test case 5: effects of parallel connections

In this test case we examine the effects of parallel connection on throughput using *simpleperf*. H1 opens two parallel connections to communicate with h4 with the -P flag, while h2-h5 and h3-h6 communicates with the normal client.

4.7.1 Results and discussion

Result:

Host pairs	H1-h4(two parallel)	H2-h5	H3-h6
Throughput	8.21Mbps 7.77Mbps	5.56Mbps	8.24Mbps

Discussion:

The expected outcome for parallel connections would be a better utilization of the available bandwidth by distributing the traffic over multiple connections. This can help to mitigate the impact of network congestion or other factors that may limit the throughput of a single connection. In this case, the total throughput of the two parallel connections between h1 and h4 should be higher than the throughput of the single connection pairs h2-h5 and h3-h6.

The measurements show that the total throughput of the two parallel connections between h1 and h4 (8.21Mbps + 7.77Mbps = 15,98Mbps) is higher than the throughput of the single connection pairs h2-h5 (5.56Mbps) and h3-h6 (8.24Mbps). This indicates that the parallel connections are effectively utilizing more available bandwidth between h1 and h4 than the single connections for the other host pairs.

When examining the throughput results of h2-h5 and h3-h6, it becomes apparent that there is a difference in the throughput between these two host pairs. This difference could be attributed to various factors that we have discussed earlier, such as network conditions, variations in network paths, or even contention for shared resources.

It is important to note that all of the results are specific to the experiment's conditions and might not be universally applicable. The throughput can vary depending on the network topology, the amount of traffic on the network, and the types of applications running on the hosts.

The effect of parallel connections in this test case, shows an improvement in throughput compared to single connection. This demonstrates the benefit of using parallel connections to better utilize available network resources and increase the overall throughput between communicating hosts.

5 Jains Fairness Index

This section of the report is meant to evaluate the resource allocation in the network system by examining the throughput allocation among competing host-pairs. JFI is a quantitative measure of fairness that ranges from 0 to 1, with 1 being the most fair and 0 being the least fair.

Furthermore, as JFI is primarily used to measure the fairness of resource allocation among competing connections that share a common network path. However, if the nodes don't share the same network path, the JFI value might not provide useful information about the network behaviour. Therefore I have decided to only measure the JFI value for specific scenarios.

The calculation methodology employed in this report to measure JFI is based on an earlier lab assignment. The results are as follows:

Test case 4:
When utilizing the jfi.py code, we get the following results:

Scenario	Jains Fairness Index
1	0.9653
2	0.9492
3	0.9254

The JFI values for each scenario suggests that the network is effectively utilizing multiplexing to distribute the available bandwidth fairly among the different network paths. However, there is a slight decrease in fairness as we move from scenario 1 to scenario 3. This could be due to various factors such as changes in traffic patterns and increased network complexity.

Despite the decreasing trend in JFI values, the overall fairness remains relatively high in all scenarios, which suggests that multiplexing is effectively managing the network resources.

Test case 5 – parallel connection:
Since h1-h4 has two parallel connections, we will treat them as separate flows. The JFI value of 0.9785 suggests that the network resources are fairly distributes among the host pairs, even with a parallel connection between h1-h4. This outcome could be a result of efficient multiplexing. It shows that the network is capable of handling multiple flows, including parallel connections, while maintaining a high level of fairness in resource allocation.

6 Conclusions

In conclusion, the various test cases conducted in this series of experiments has provided valuable insights into different aspects if network performance, including latency, throughput, multiplexing and parallel connections. The tests demonstrated the impact of network topology, congestion, and bottlenecks on communication between host pairs.

The measurements showed that latency and throughput can be significantly affected by the network conditions and the sharing of network resources. The experiments involving multiplexing illustrated that as several host pairs communicate simultaneously, the overall network performance might be degraded due to increased congestion.

Additionally, the test involving parallel connections revealed that while it is possible to improve throughput by using parallel connections, the improvement will likely not be linear and could still be limited by shared resources and network conditions.

Throughout the test cases, a recurring theme was the importance of understanding the specific network conditions and the impact of various factors on network performance. By examining these results, one could make informed decisions to optimize network configurations, address potential bottlenecks, and improve overall network performance.

7 References (Optional)

Lamberti, A. (2023, 21. Feb) What Is Packet Loss: The Invisible Enemy of Network Performance, Obkio. https://obkio.com/blog/what-is-packet-loss/

Adiyatri (2021, 10. Sep) What is Ping? https://www.geeksforgeeks.org/what-is-ping/

Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, Kaustubh Prabhu, https://iperf.fr/

TCP vs UDP Comparisonhttps://www.youtube.com/watch?v=uwoD5YsGACg

GeeksForGeeks (2023, 6. April) Layers of OSI model: https://www.geeksforgeeks.org/layers-of-osi-model/

Gillis, S. A. (2020, January) Latency https://www.techtarget.com/whatis/definition/latency

Gillis, S. A. (2021, August) Packet Loss
https://www.techtarget.com/searchnetworking/definition/packet-loss

Indeed Editorial Team (2023, 27. Jan) Network Throughput vs. Bandwidth: Differences and Tools
https://www.indeed.com/career-advice/career-development/throughput-vs-bandwidth