CN Assignment 3 Report

Srimant Mohanty
2021207
srimant21207@iiitd.ac.in

Plots

The experiment involved capturing packets exchanged between the client and the server on the server's interface and subsequently processing the acquired PCAP files using Python scripts. Two key metrics: the Average Throughput for each TCP flow (measured in bits per second) and the Average Latency for each TCP flow (measured in milliseconds) were computed to assess the performance of various TCP server programs that we developed as part of our experiments.

The following plots were obtained for 500, 1000, and 3000 simultaneous connections for each experiment.

Observations

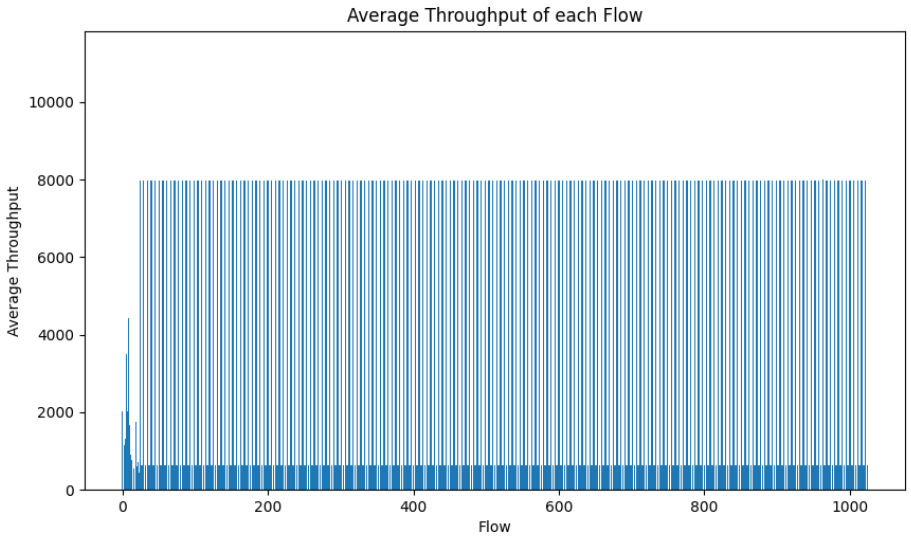
Throughput

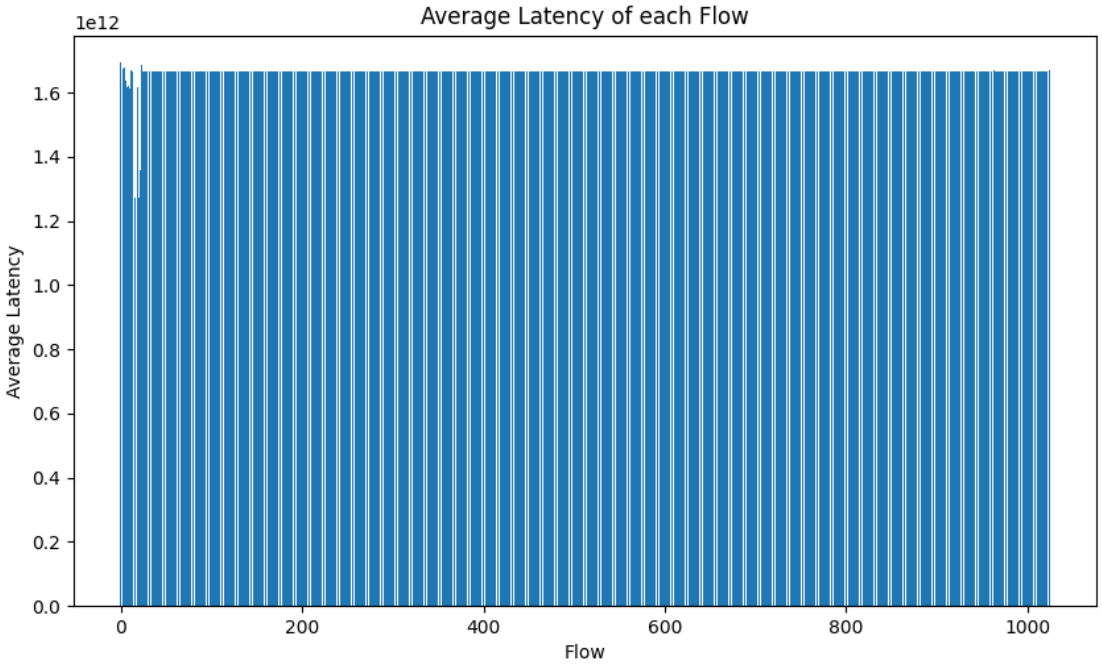
- The plots represent average throughput in bits per second and the average latency is milliseconds for each TCP flow during each scenario/experiment. The X-axis represents the different TCP flows which may be uniquely identified by a 4-tuple Source IP Address, Source Port Number, Destination IP address and Destination Port Number.
- The plots show that the average throughput values remain relatively consistent across all the 5 experiments but with an increasing number of TCP connections the throughput slightly decreases i.e. from 500 to 1000 to 3000 Connections.
  This can be especially observed in cases of Select, Poll & Epoll System Calls.
- Also it maybe observed in some scenarios that throughput increases initially on increasing number of clients however after a point the network may become congested and throughput decreases

Latency

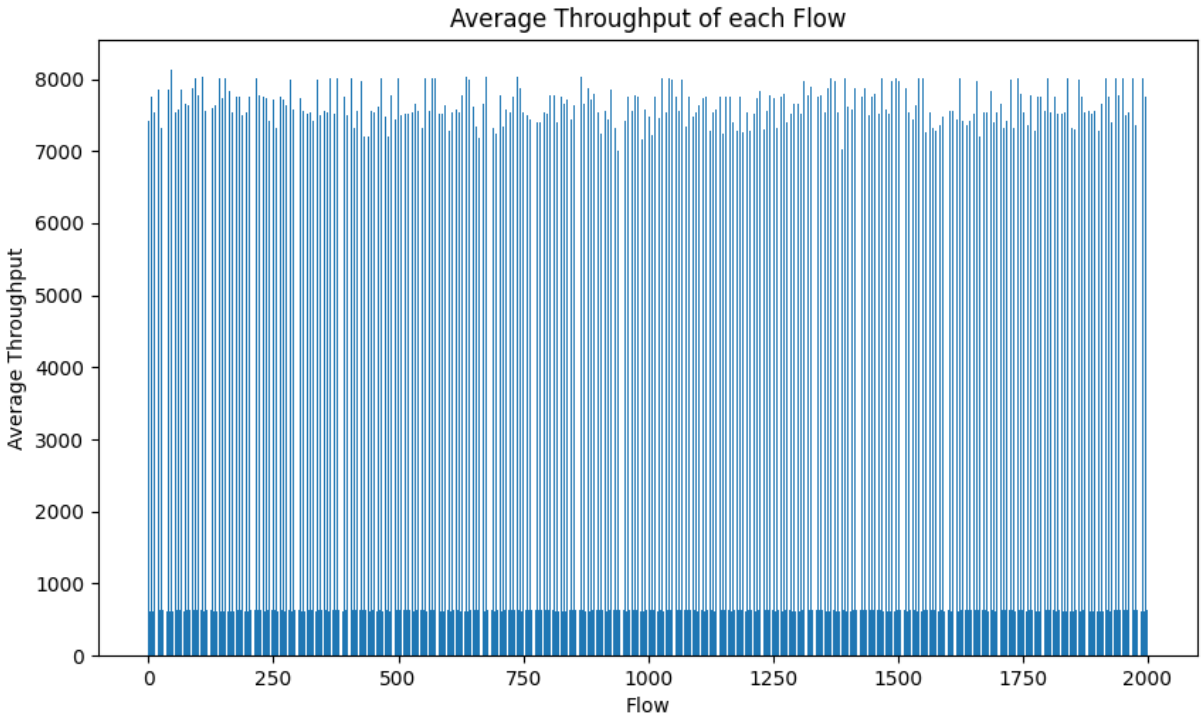
- The graphs plotted tend to show that average latency is consistent and constant with increasing number of concurrent connections and with different I/O multiplexing techniques.
- However, latency also tends to increase with an increasing number of clients in a few cases as the server has to manage more connections simultaneously, leading to longer response times for individual clients.
- I/O multiplexing techniques like select, epoll and poll are comparatively faster than traditional methods like forking and threading for handling concurrent clients.

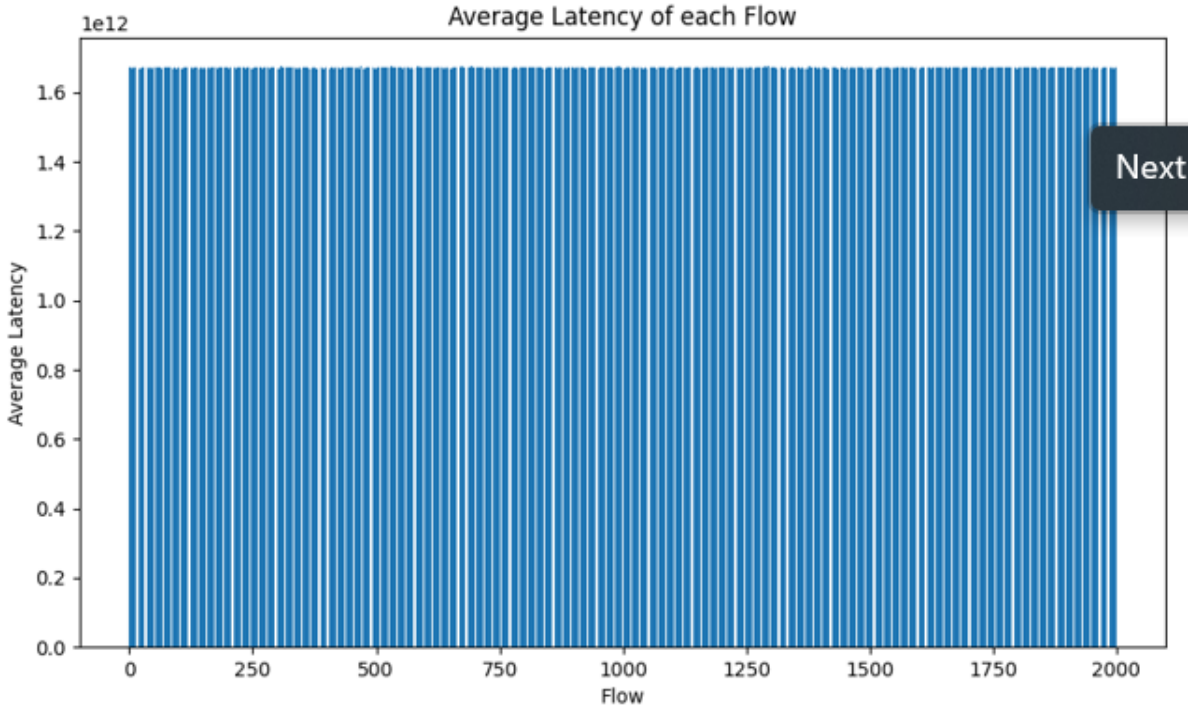
Fork 500 Connections



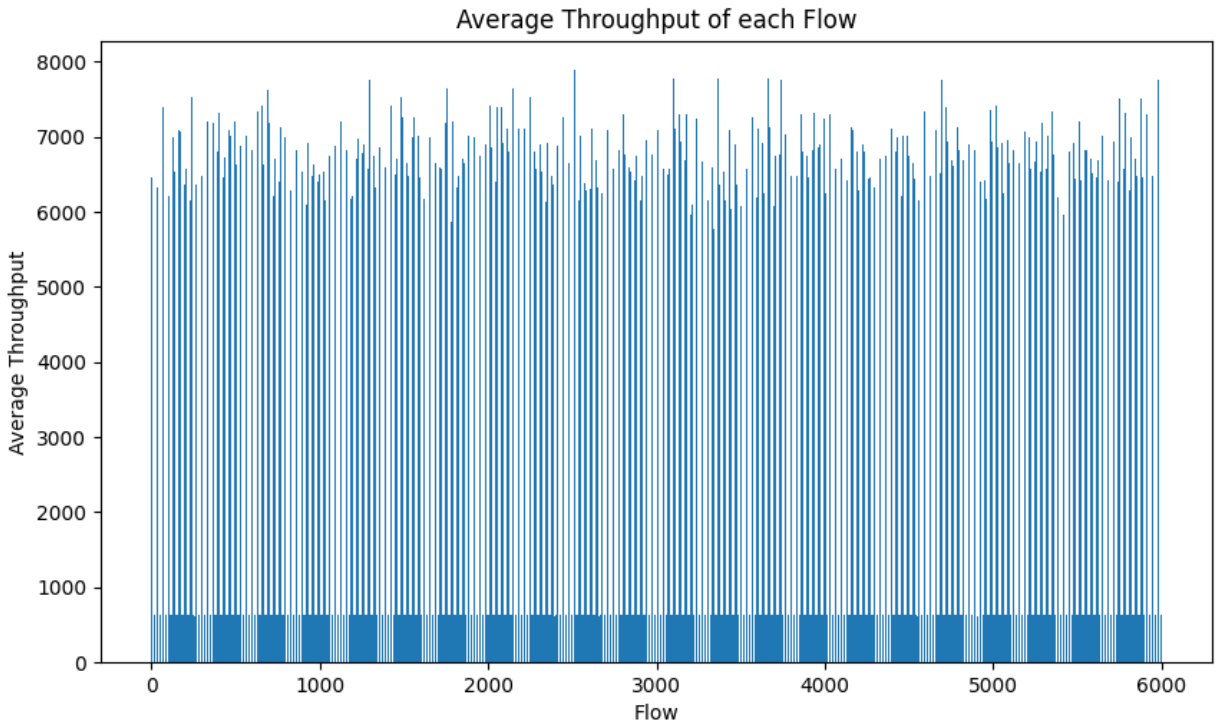


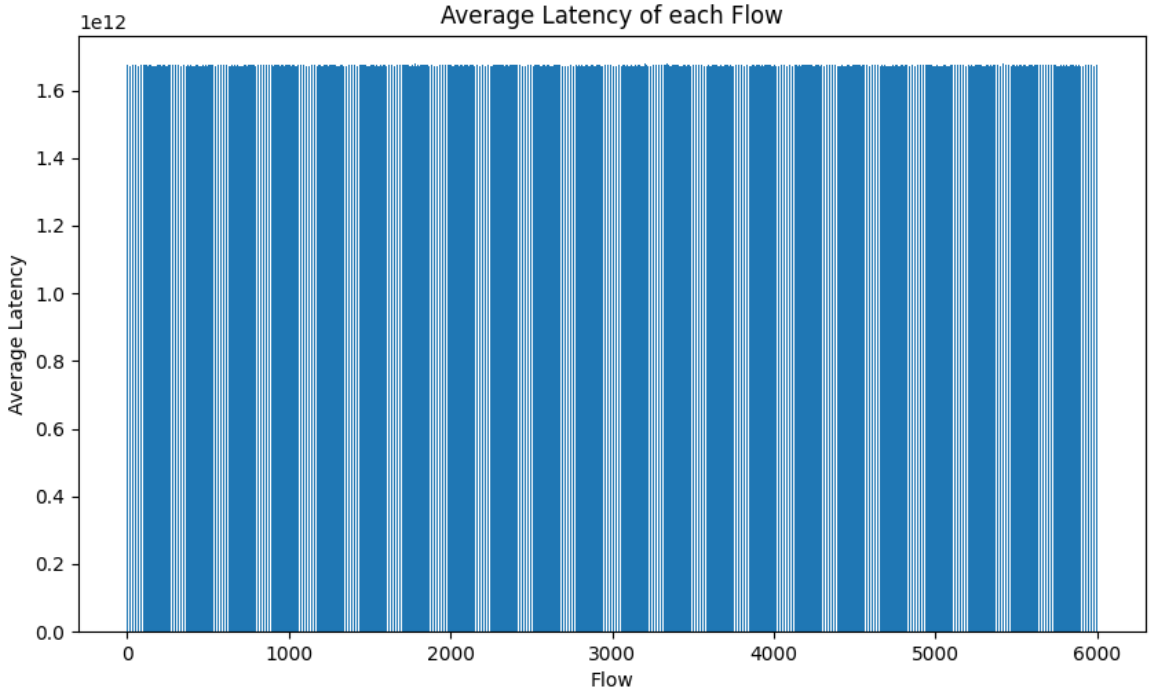
Fork 1000 Connections



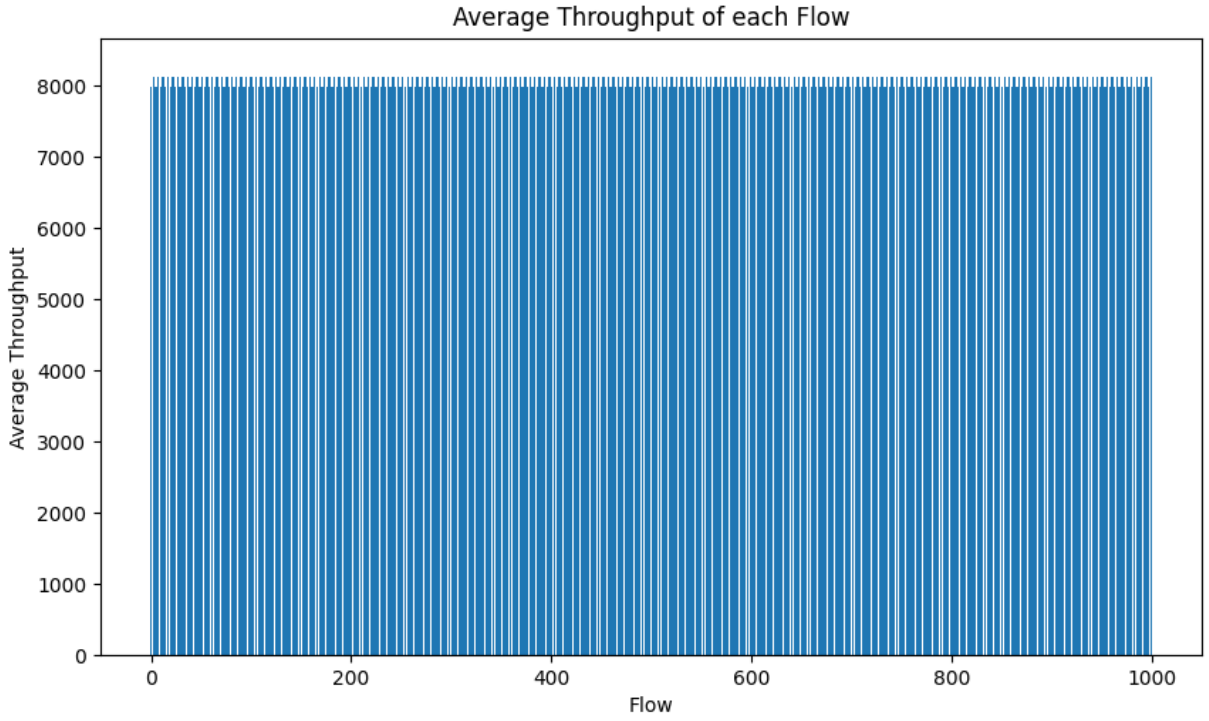


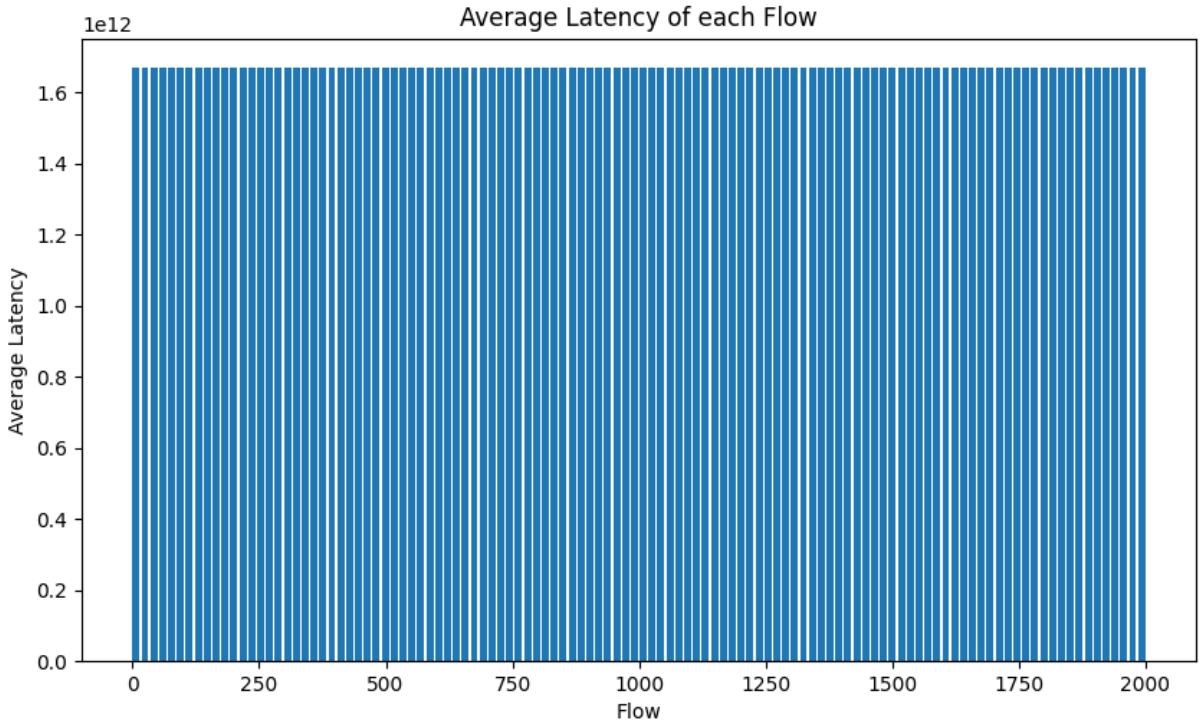
Fork 3000 Connections



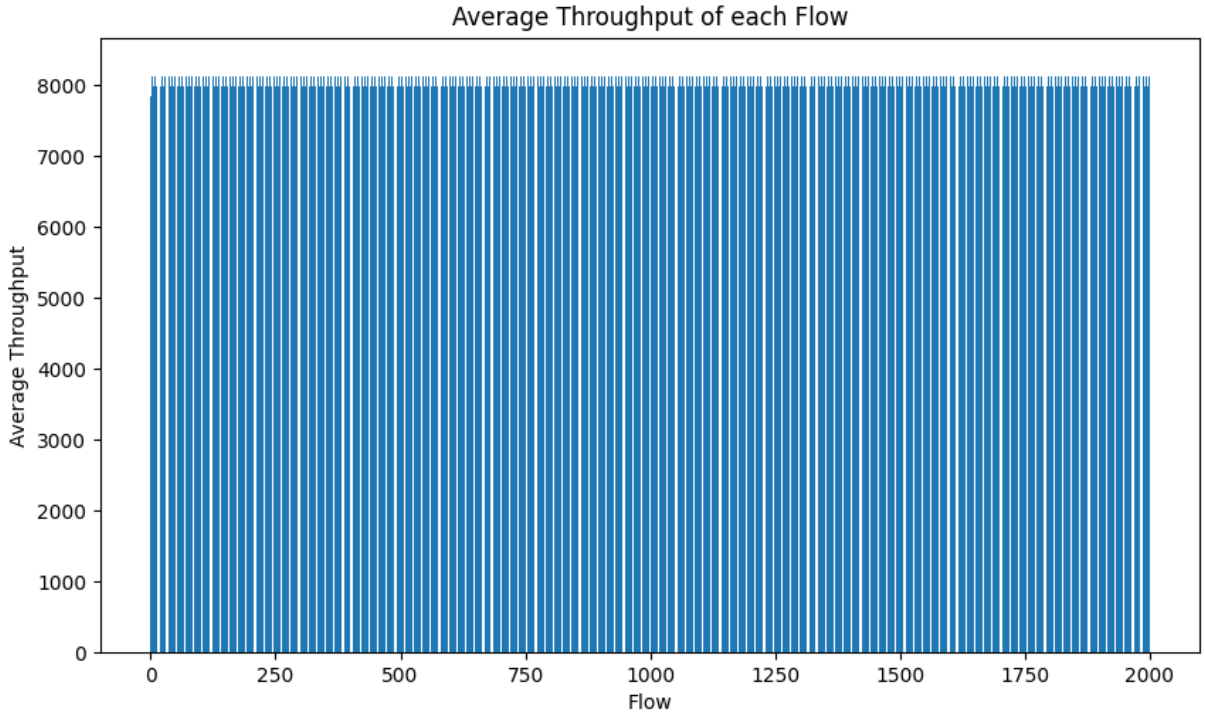


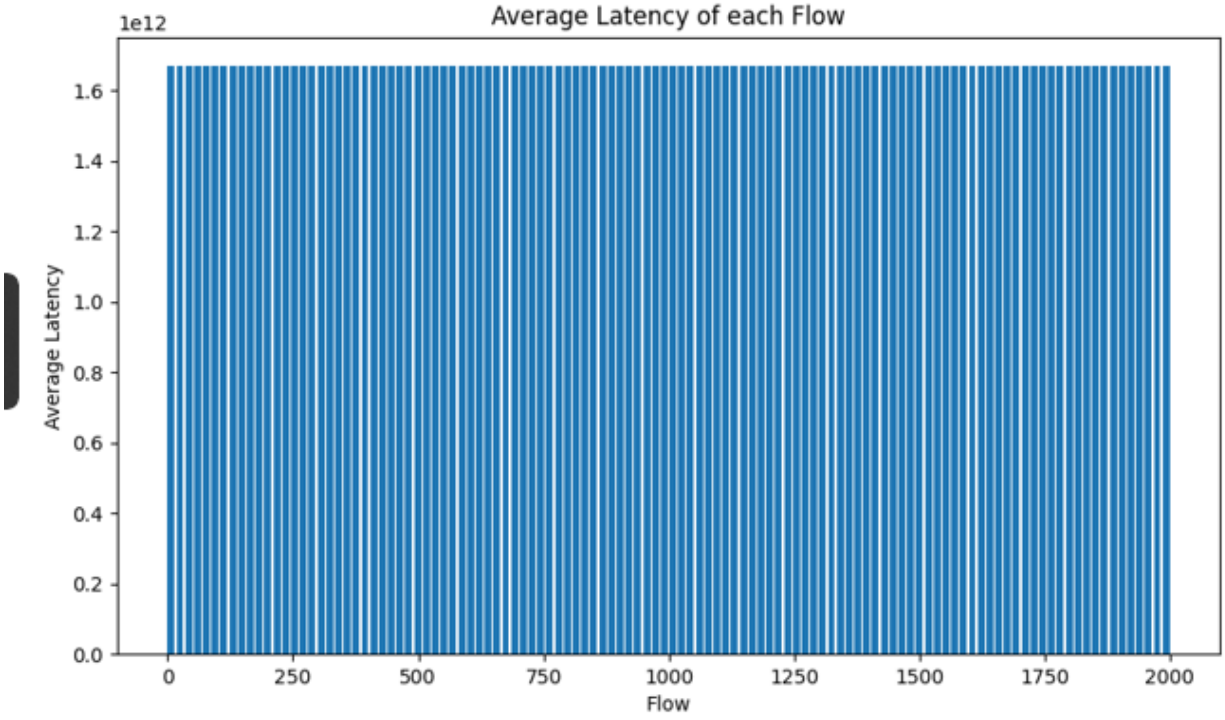
Thread 500 Connections



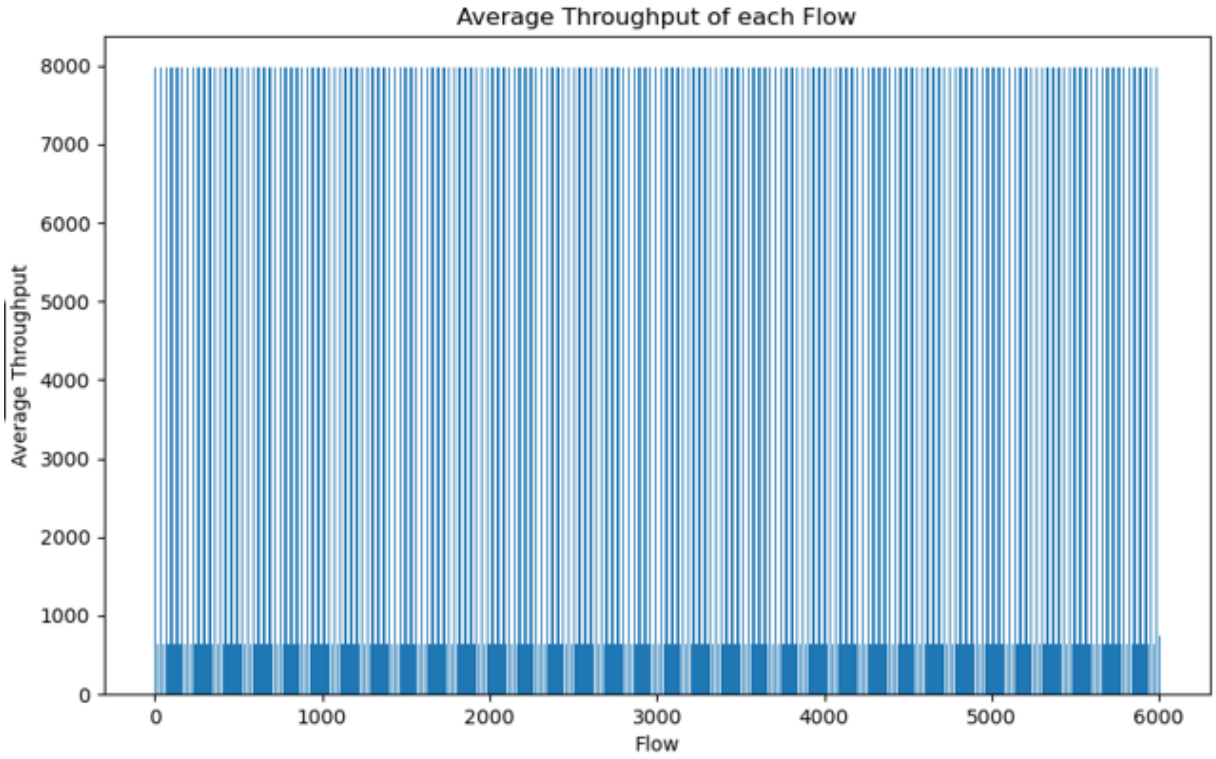


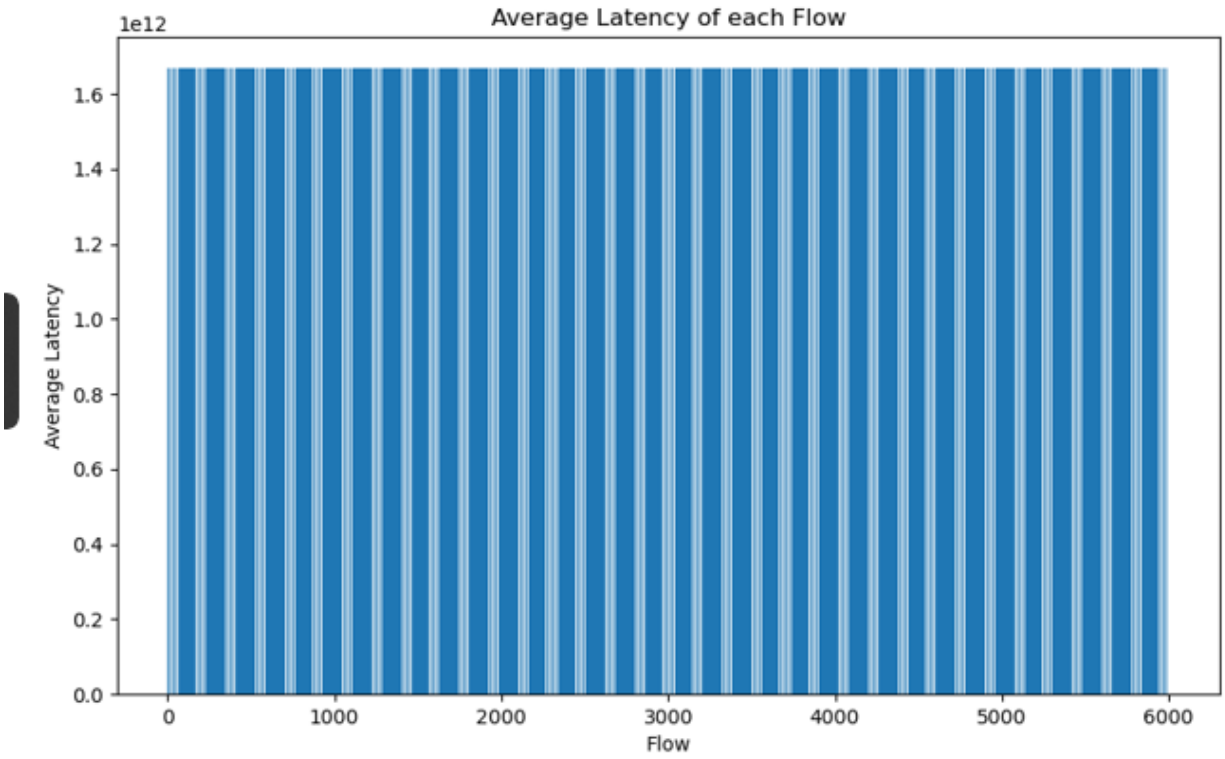
Thread 1000 Connections



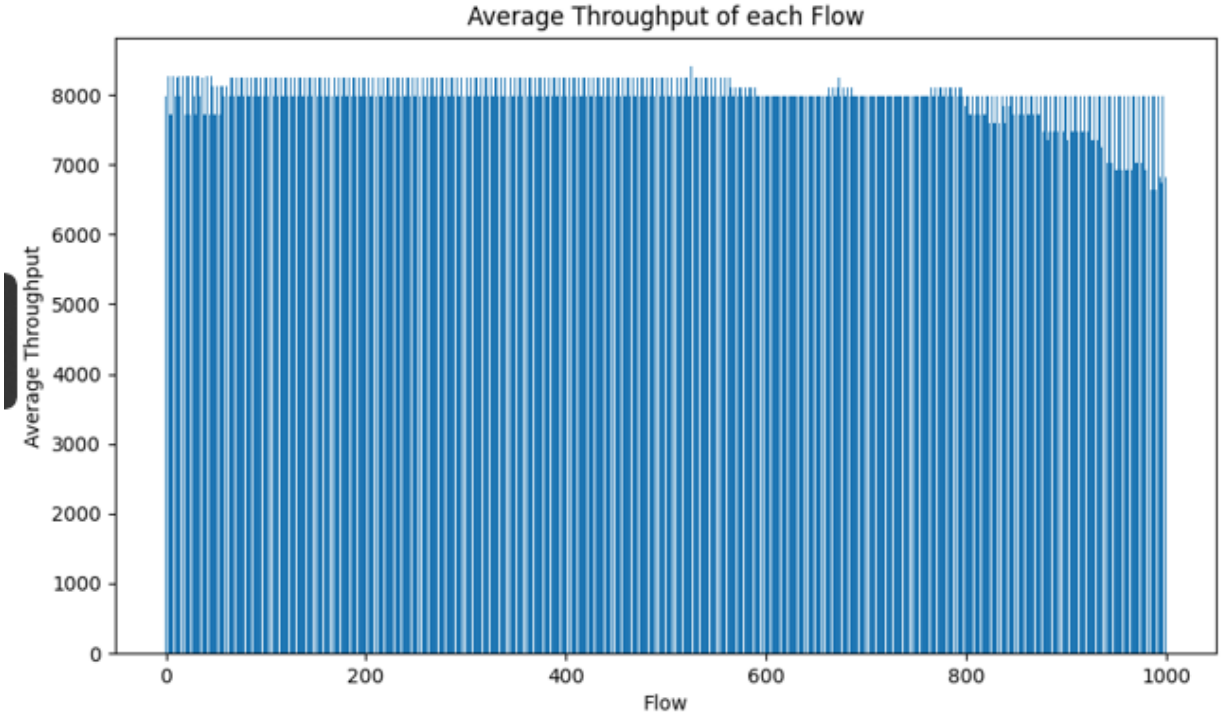


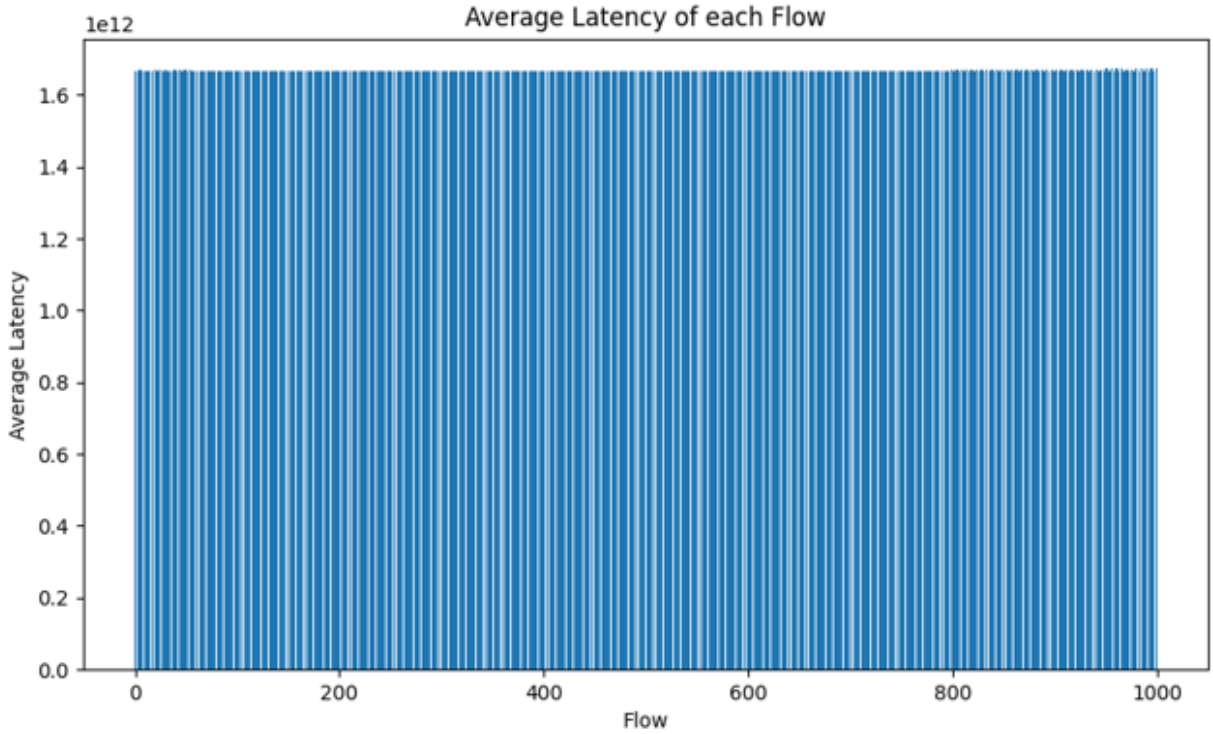
Thread 3000 Connections


Average Throughput of each Flow
Average Throughput
Flow
0
1000
2000
3000
4000
5000
6000
7000
8000

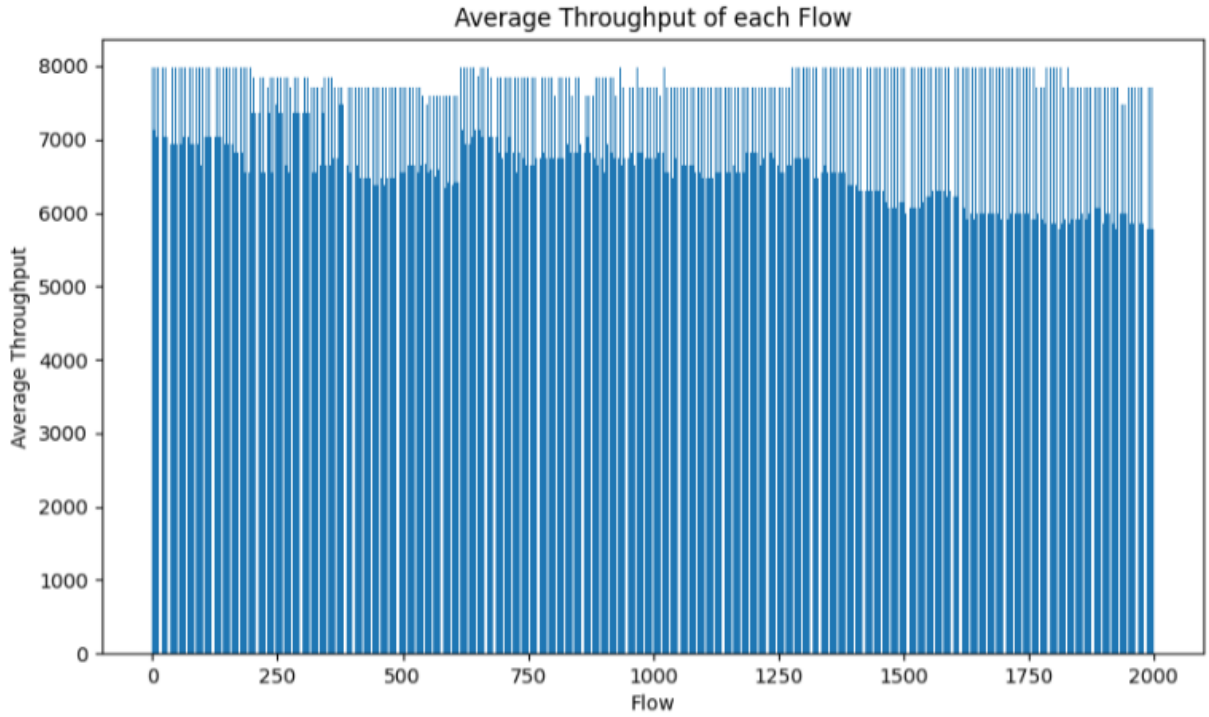

Average Latency of each Flow
1e12
Average Latency
Flow
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
1000
2000
3000
4000
5000
6000

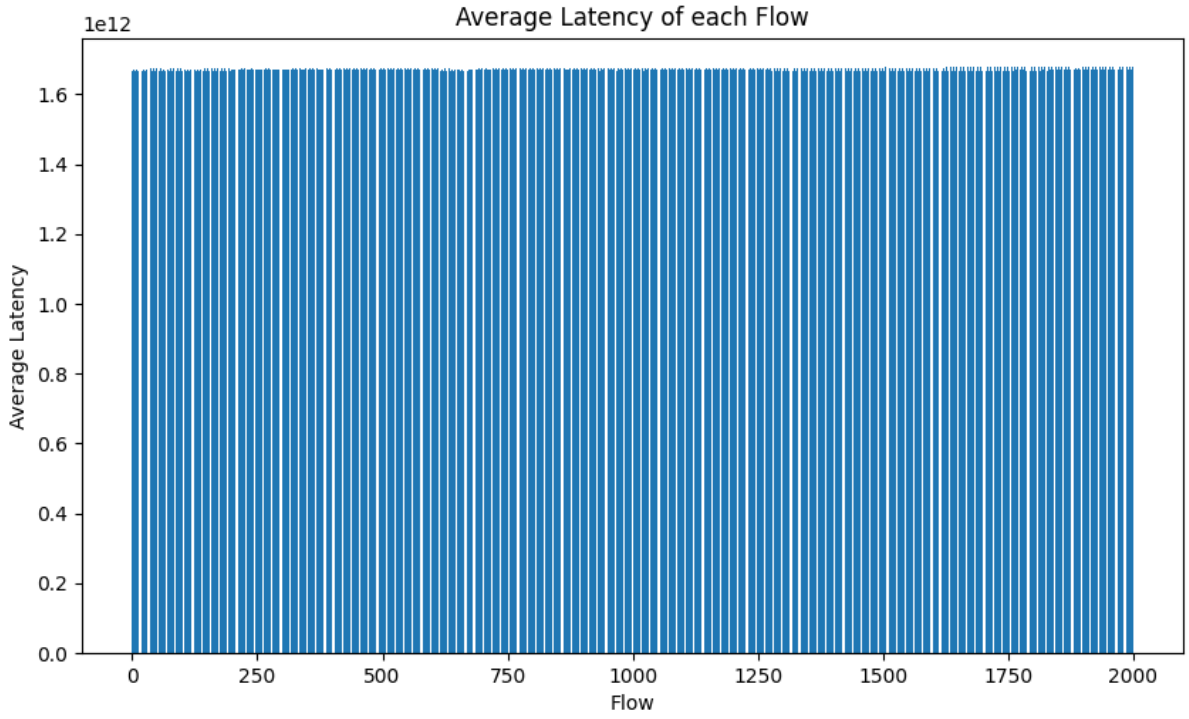
Select 500 Connections



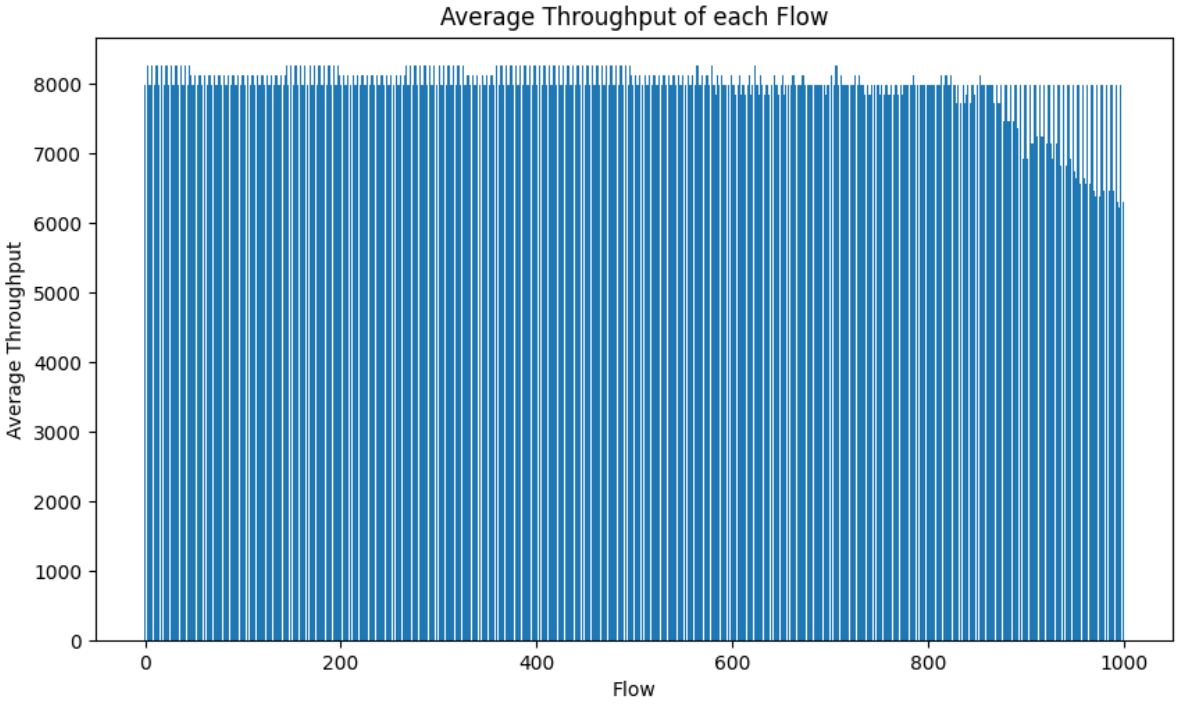


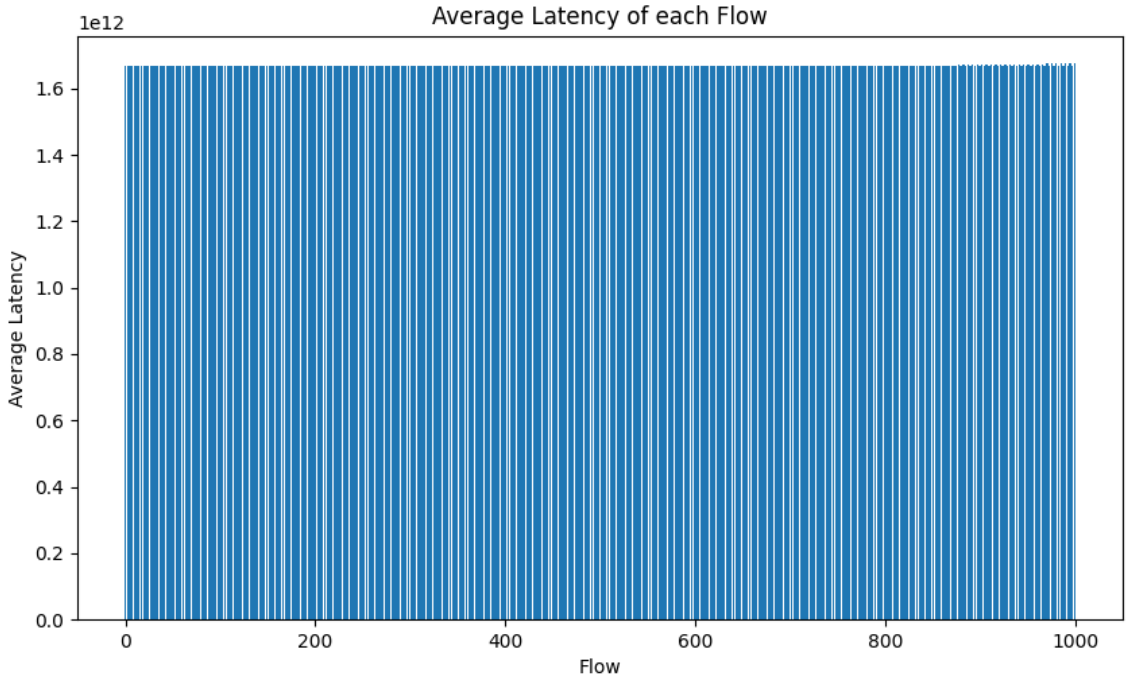
Select 1000 Connections



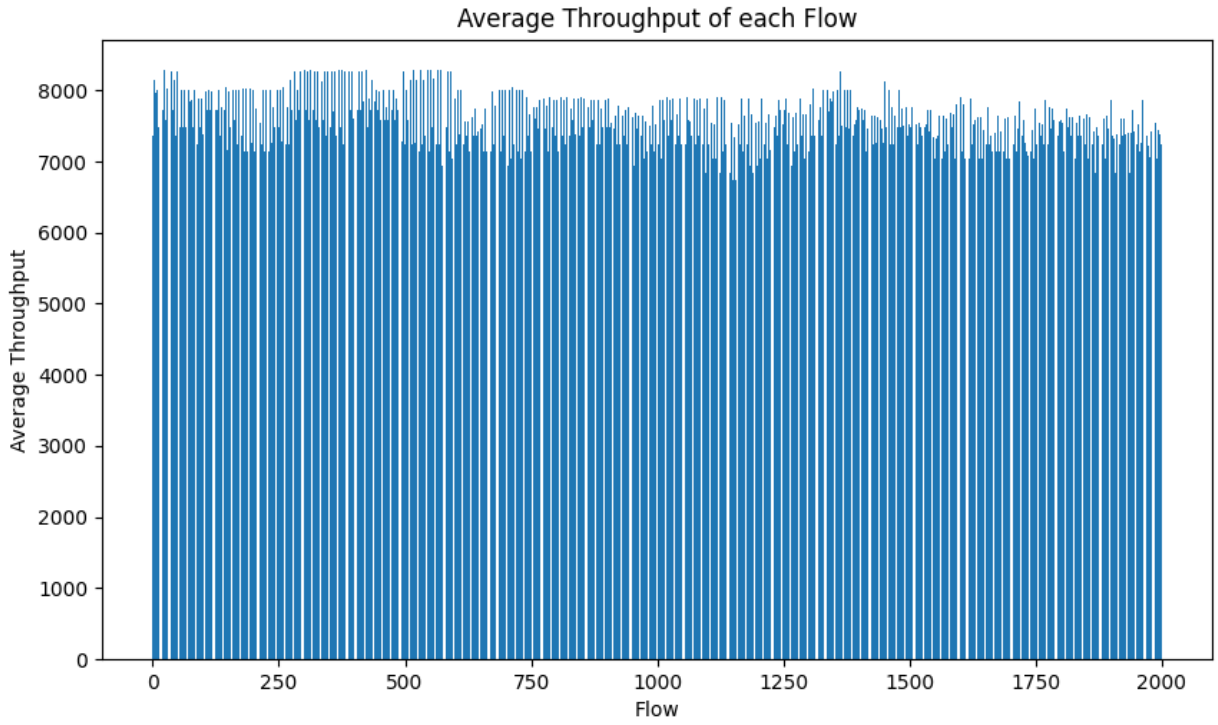


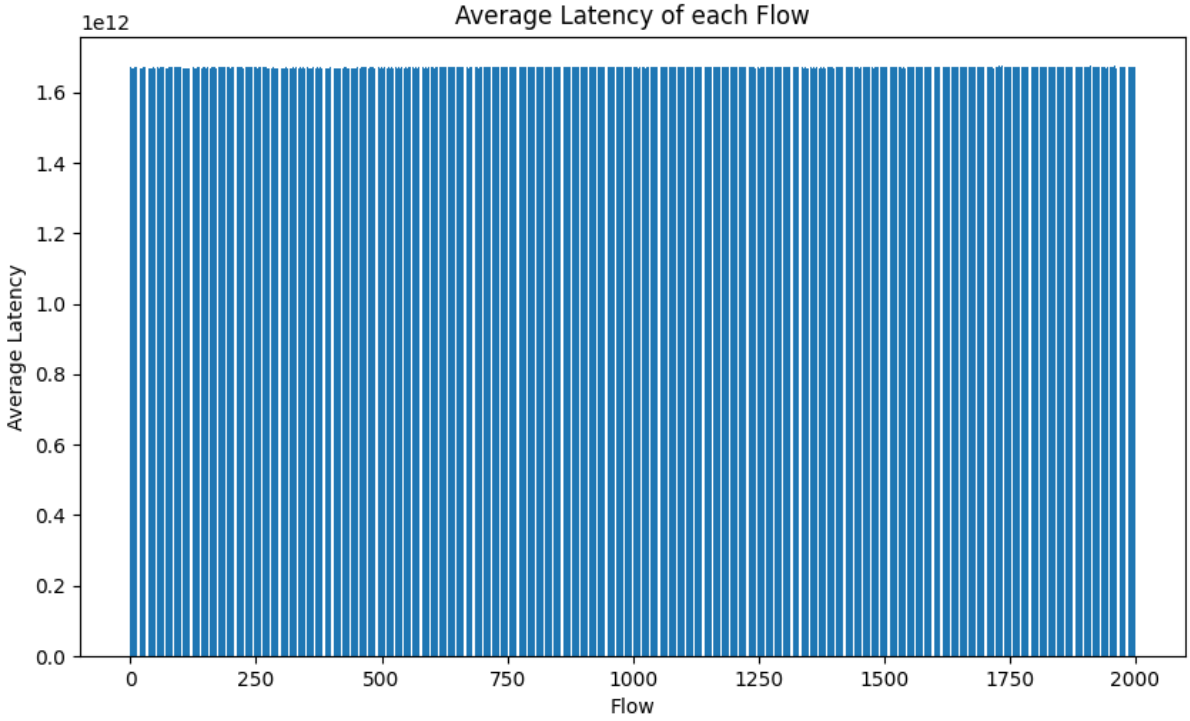
Poll 500 Connections



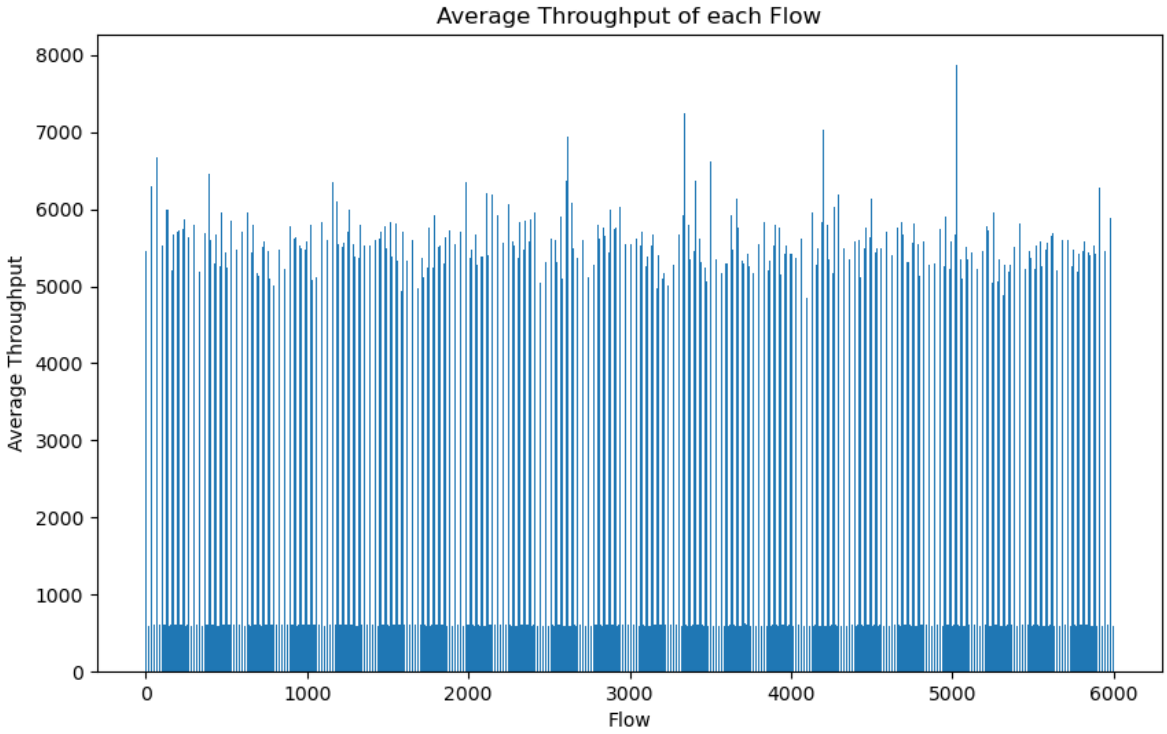


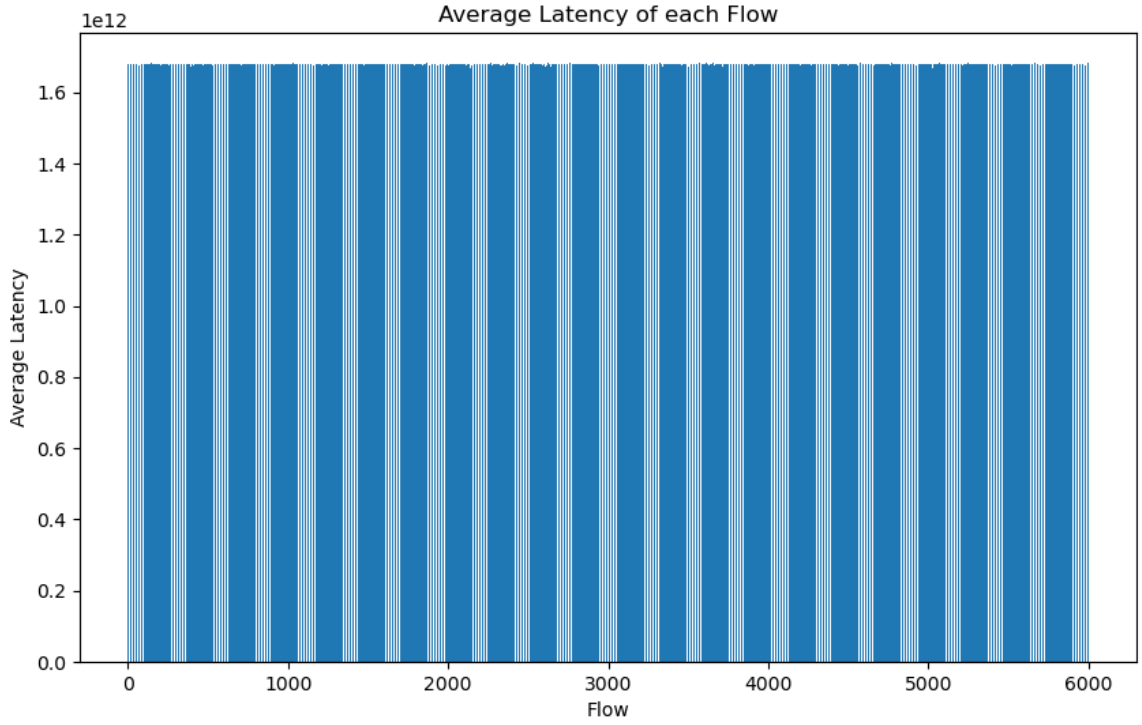
Poll 1000 Connections



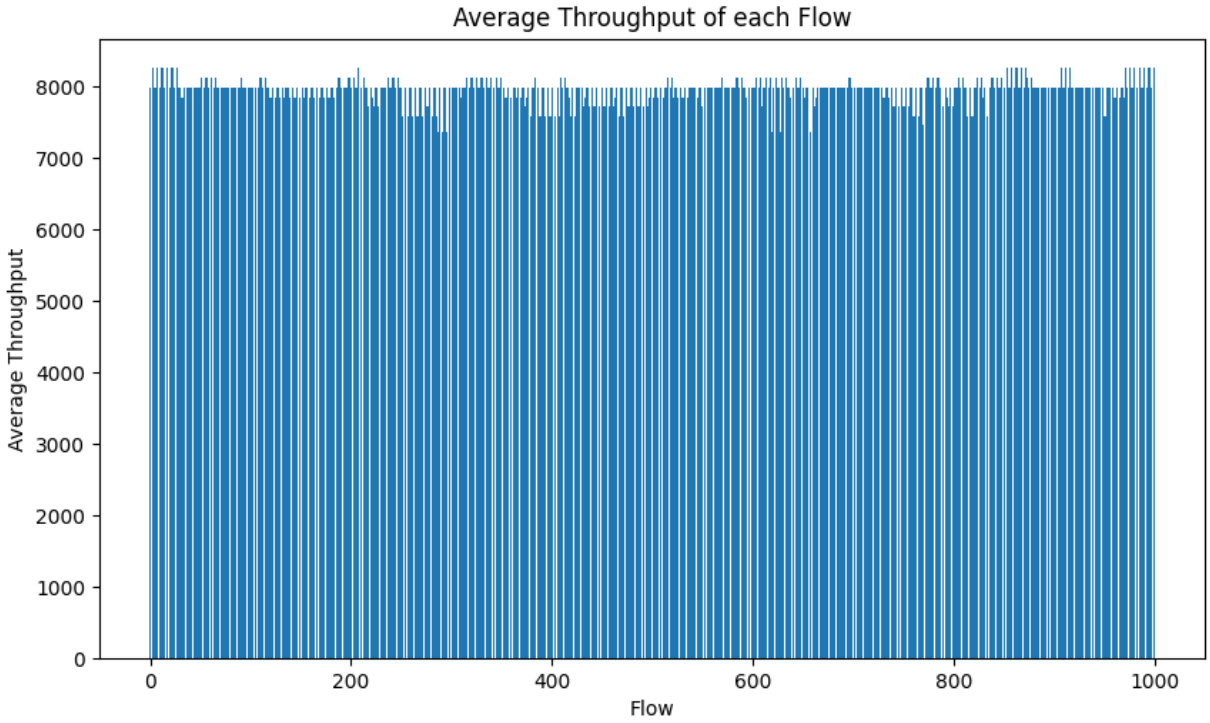


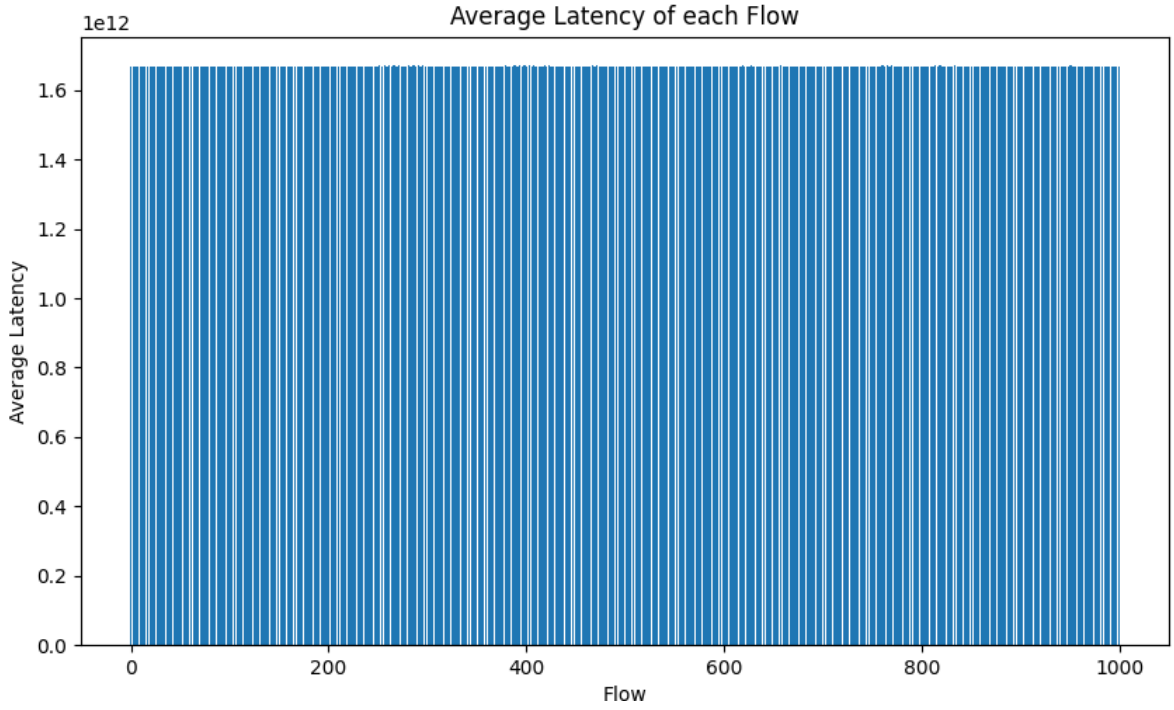
Poll 3000 Connections



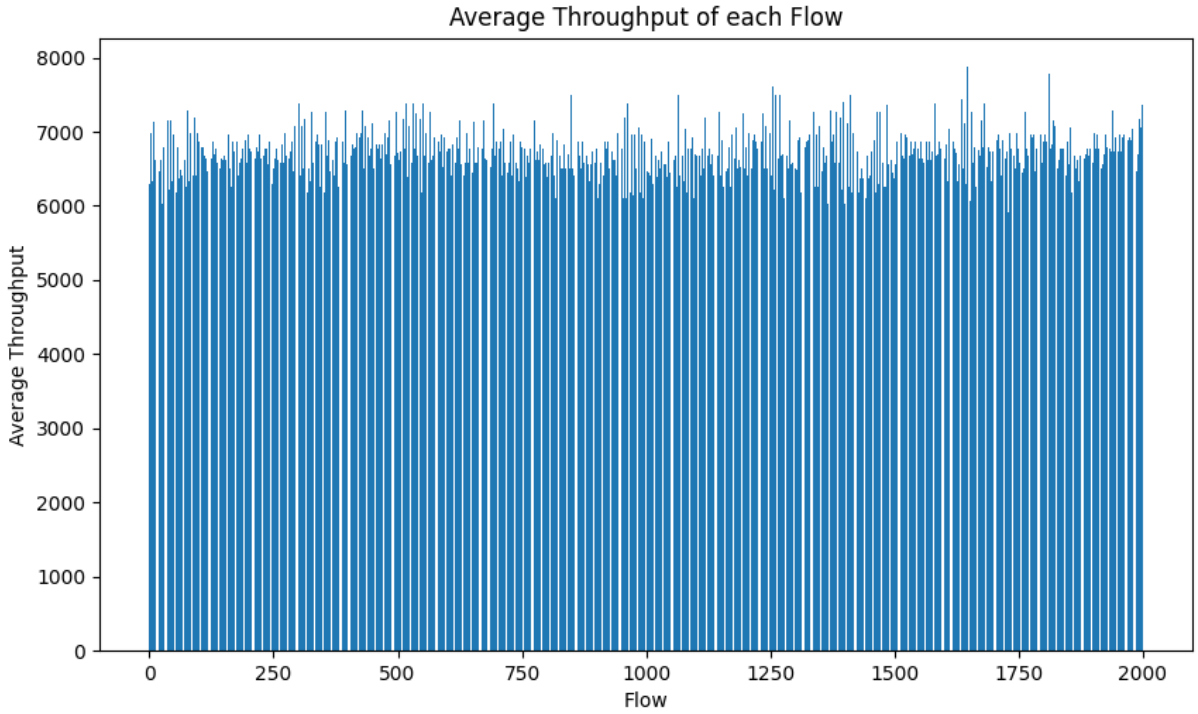


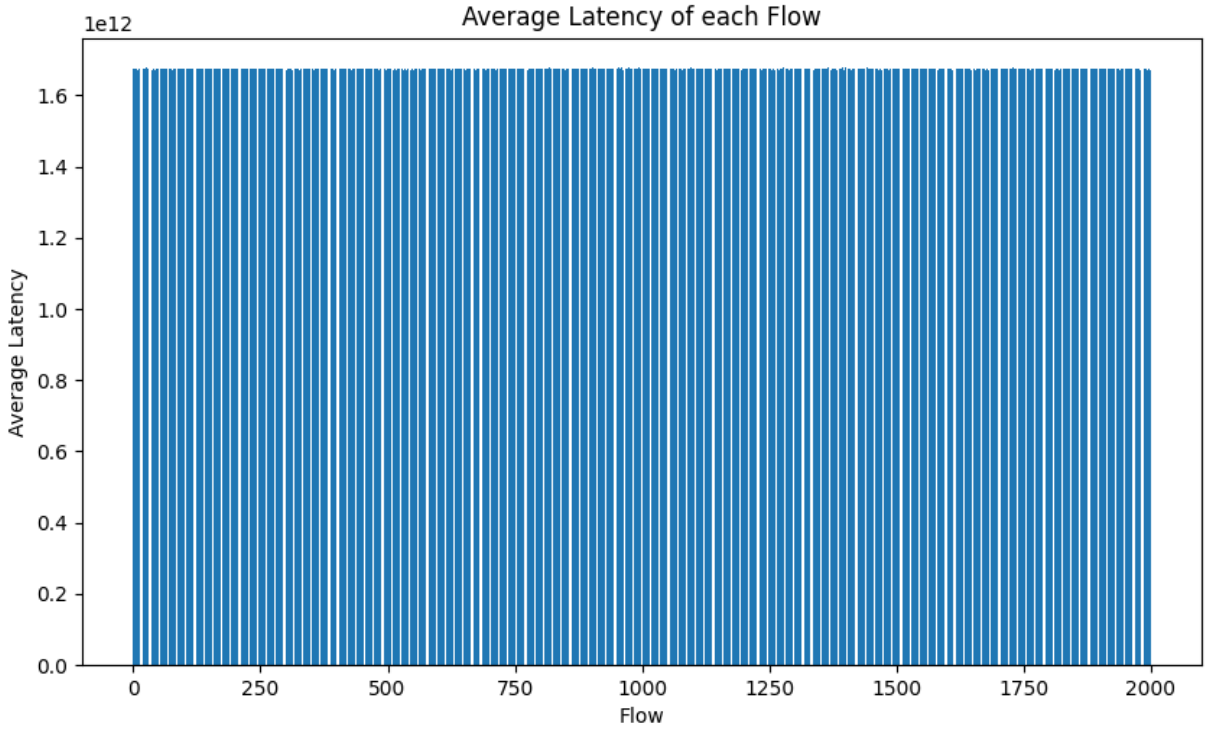
Epoll 500 Connections



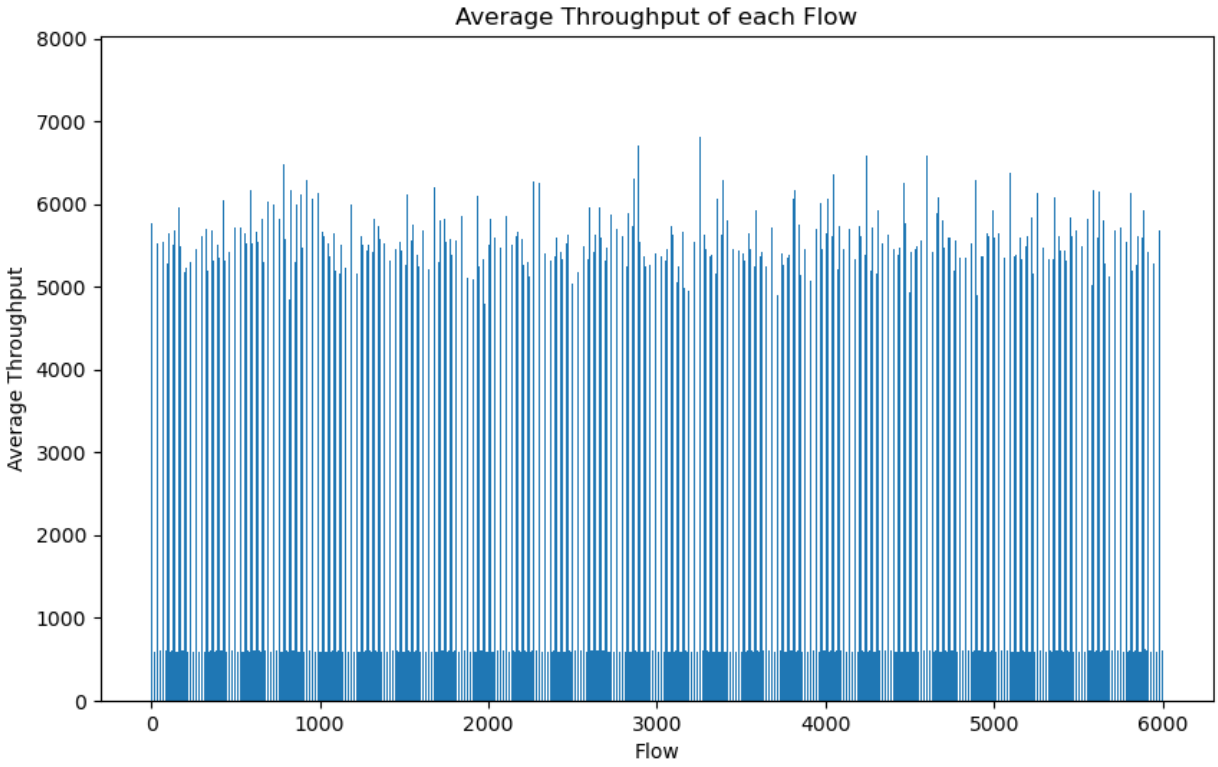


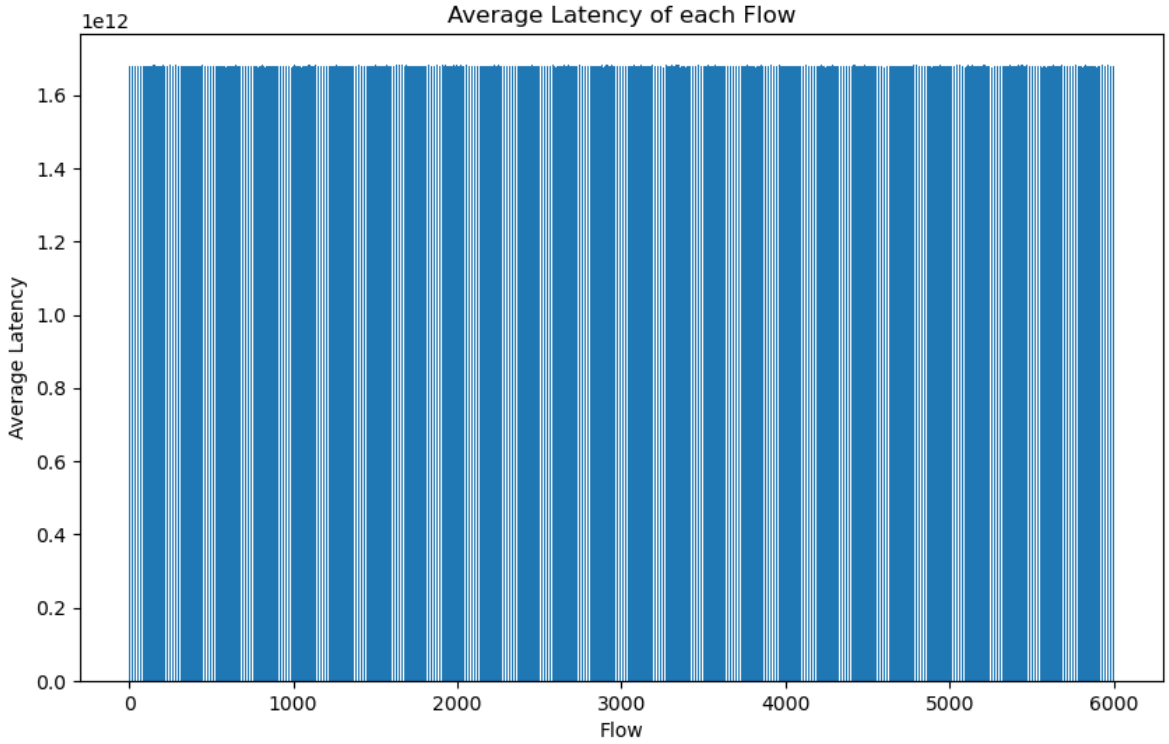
Epoll 1000 Connections





Epoll 3000 Connections





Server process's CPU utilization

Process	CPU Utilization(%)
No Process	**0.7%**
Fork 500 Connections	**11.7%**
Fork 1000 Connections	**49.8%**
Fork 3000 Connections	**66.5%**
Thread 500 Connections	**17.7%**
Thread 1000 Connections	**29.1%**
Thread 3000 Connections	**36.2%**
Select 500 Connections	**89%**
Select 1000 Connections	**103%**
Poll 500 Connections	**92.6%**
Poll 1000 Connections	**70.7%**
Poll 3000 Connections	**91%**
EPoll 500 Connections	**15.2%**
EPoll 1000 Connections	**42%**
EPoll 3000 Connections	**93.7%**

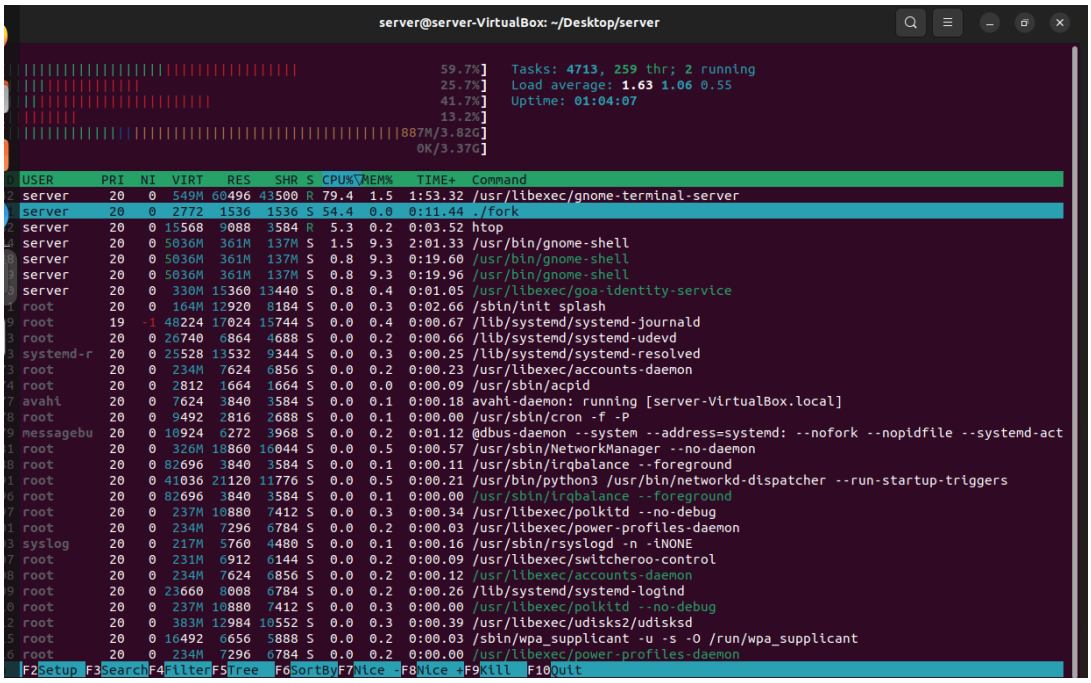
CPU utilization essentially represents the portion of a system's processing capacity that is currently in use.

Observation:

- A high CPU Utilization indicates that there is heavy load on the system due to the presence of multiple connections or multiple concurrent clients.
- We also observe that the CPU utilization values keep on varying during the period when a process is executing and the CPU Utilization % shown are those that occur more frequently or at random occurrences.
- The CPU Utilization % increases from 500 to 1000 and 3000 connections is every experiment indicating that more CPU power is required to serve multiple concurrent clients/requests.
- Different processes behave differently with respect to CPU Utilization time; for instance it drastically increases over time for a fork as more processes are being created in a fork. Similarly, I/O multiplexing techniques are more efficient as compared to fork and thread for CPU Utilization as their values are more or less stable.

- In some cases CPU Utilization may be more than 100 indicating that more than 1 core is being used.

CPU Utilization was captured in the following manner:-



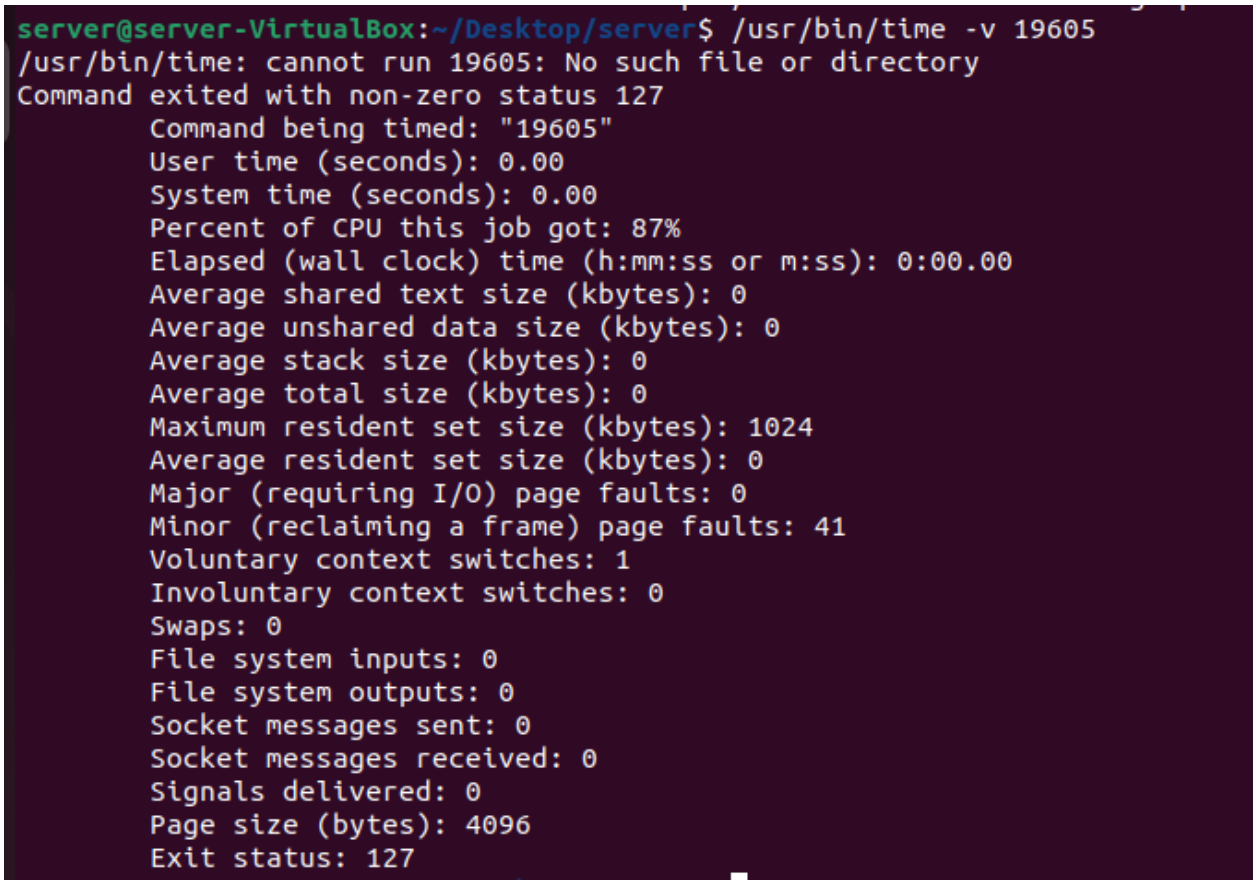
Server process's memory utilization

For understanding the server's memory usage we observe the Maximum resident size which indicates the peak memory usage of the process.

We can see that the MRSS follows a stable and consistent pattern across all the experiments indicating that there is a stable and fixed memory utilization throughout all connections.

Process	Maximum resident size(K Bytes)
Fork 500 Connections	**3328**
Fork 1000 Connections	**3456**
Fork 3000 Connections	**3456**
Thread 500 Connections	**3456**
Thread 1000 Connections	**3456**
Thread 3000 Connections	**3328**
Select 500 Connections	**3456**
Select 1000 Connections	**3456**

Poll 500 Connections	**3456**
Poll 1000 Connections	**3328**
Poll 3000 Connections	**3456**
EPoll 500 Connections	**3456**
EPoll 1000 Connections	**3456**
EPoll 3000 Connections	**3456**



References

https://support.cloudways.com/en/articles/5120765-how-to-monitor-system-processes-using-htop-command

https://www.geeksforgeeks.org/using-htop-to-monitor-system-processes-on-linux/

https://www.softprayog.in/programming/io-multiplexing-select-poll-epoll-in-linux