Module code: CS4105
Matriculation number: 200017941

CS4105 Practical P1 - TCP performance

Introduction

For this coursework, I conducted an experiment to investigate the performance of TCP when the delay is increased. As part of the experiment, the measurements made were stored in CSV files. These results were then processed by performing simple statistical calculations. Additionally, the results were used to plot graphs to show how the performance of TCP changes. All the measurements were collected alone.

Methodology

To generate TCP flows between two hosts within the network, iperf(1) was used and to add the necessary delay tc(1)/netem(8) was used. The chosen host names are pc7-100-l which is the client and pc7-011-l which is the server.

The experiment was ran through the use of bash scripts to automate all the necessary tasks. The results were stored in the form of a CSV file, and using Python (pandas, matplotlib), the statistical analysis and plotting of the data was done.

The end-to-end throughput (Mbits/sec) is what is mainly being measured to evaluate the performance of TCP. Additionally, the amount of data transferred (Gbytes) and initial round trip time (ms) were also being measured. All these acts as the independent variables for the experiment. The delay is the dependent variable and is therefore changed to see its effect on the independent variables. The time interval (10 sec) , congestion window (14 bits) and maximum segment size (1448 bits) serve as the control variable and so are kept the same throughout the experiment.

As for the design, a total of five different types of experiments were conducted, where each type involves a series of TCP flows being sent from the client to the server and is characterised by how the delay is changed between each of these TCP flows. Each type is describes as follows:

- No delays added per TCP flow (delay = 0).
- A constant delay added per TCP flow (delay = c).
- A linear increase in the delay added per TCP flow (delay = x + 5).
- A quadratic increase in the delay added per TCP flow (delay = x^2).
- A exponential increase in the delay added per TCP flow (delay = 2^x).

In general, the experiments can be split into three categories: one where there is no delay, one where the delay is constant for every transfer and one where the delay is increased after each transfer which can be represented in the form of an equation (as stated above). For the first two categories, a total of ten transfers were made so that the average of the given delay can be calculated to improve the reliability of the experiment. As for the third category, the transfers were continuously made until the maximum allowed limit for the delay is reached, which in this case is 64ms, instead of having a fixed number. Furthermore, for the experiment with a constant delay, a series of delays

between 0ms – 60ms was used to show how TCP performs within the allowed range of delay. Each type of experiment has its own bash script and CSV file.

The main focus of this experiment is to see how TCP performs when there is no delay, and for this to be compared to its performance across varying levels of constant delay (stated in the first two bullet points above). As a result, this would give us a picture of how TCP's performance changes the worse the delay gets. Therefore, descriptive statistics and analysis of the graphs will mainly focus on this. However, the experiment is further extended and measurements were also taken for cases where the delay is constantly increasing at a certain rate. I chose to incorporate this in my experiment firstly out of curiosity, but secondly with the intent of replicating different scenarios that may occur within the network, in terms of not only the level of delay itself, but also the rate of change of the level of delay. This is because the state of the network can be vary tremendously and is often times unpredictable, which means that delay is unlikely to be a constant for a series of transfers.

Results

TCP flows with no delay

Measurements table:

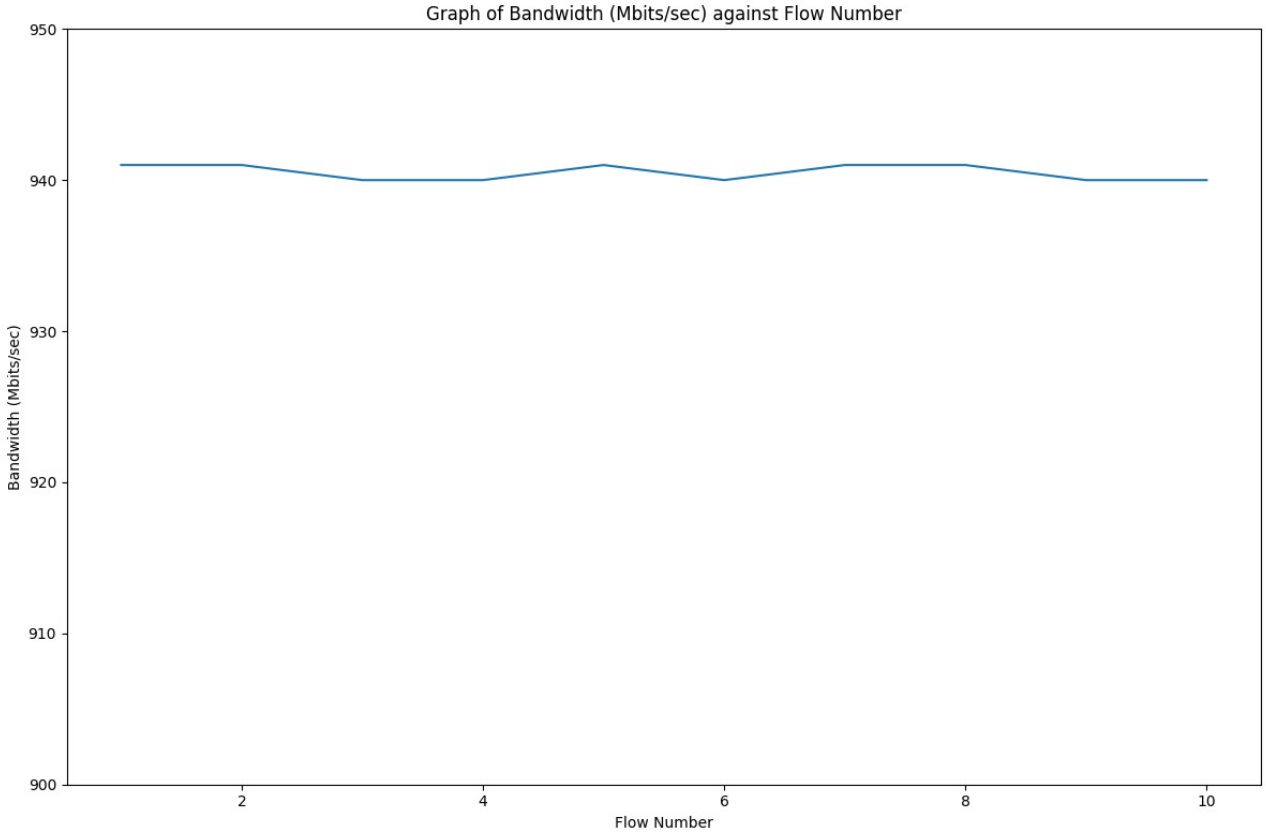
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	0.0	1.1	941.0	256.0
1	0.0	1.1	941.0	163.0
2	0.0	1.1	940.0	160.0
3	0.0	1.1	940.0	221.0
4	0.0	1.1	941.0	135.0
5	0.0	1.1	940.0	251.0
6	0.0	1.1	941.0	95.0
7	0.0	1.1	941.0	66.0
8	0.0	1.1	940.0	94.0
9	0.0	1.1	940.0	168.0

Module code: CS4105
Matriculation number: 200017941

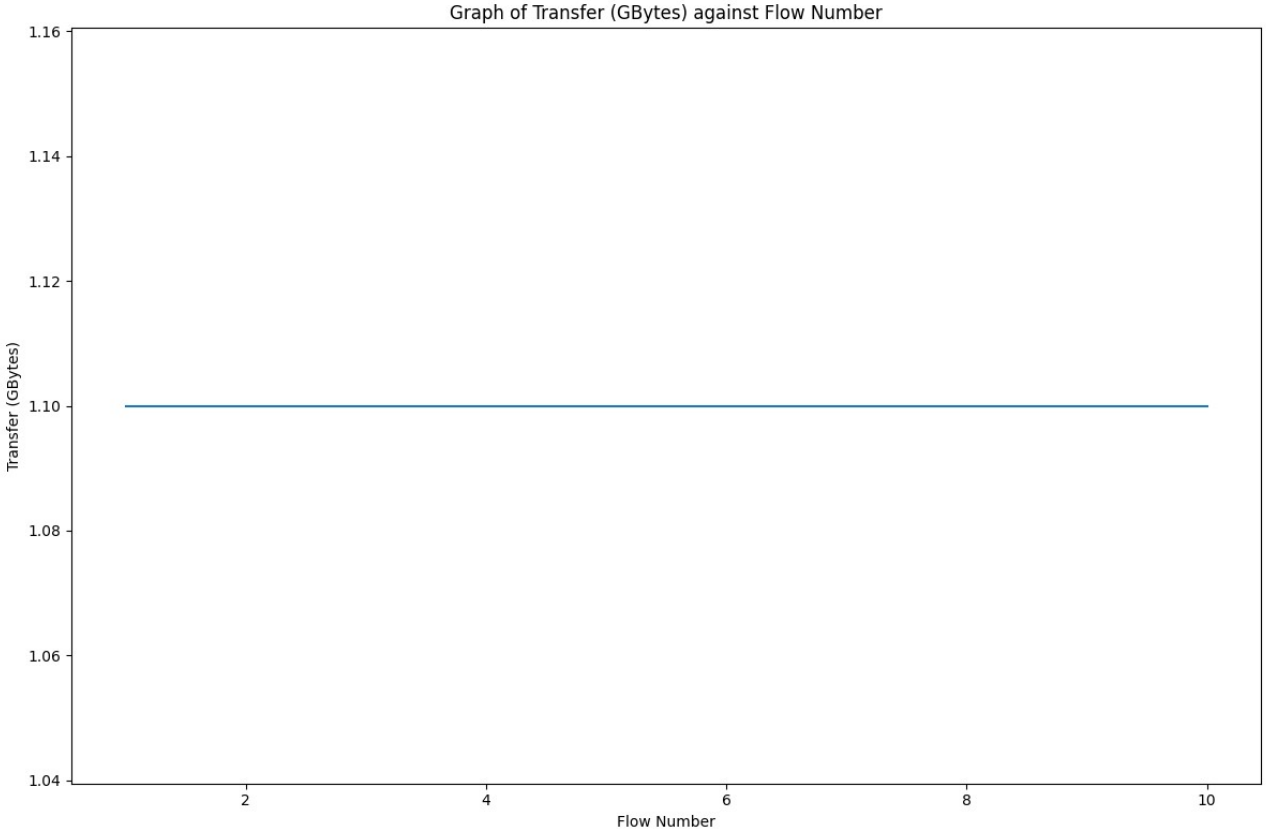
Descriptive statistics:

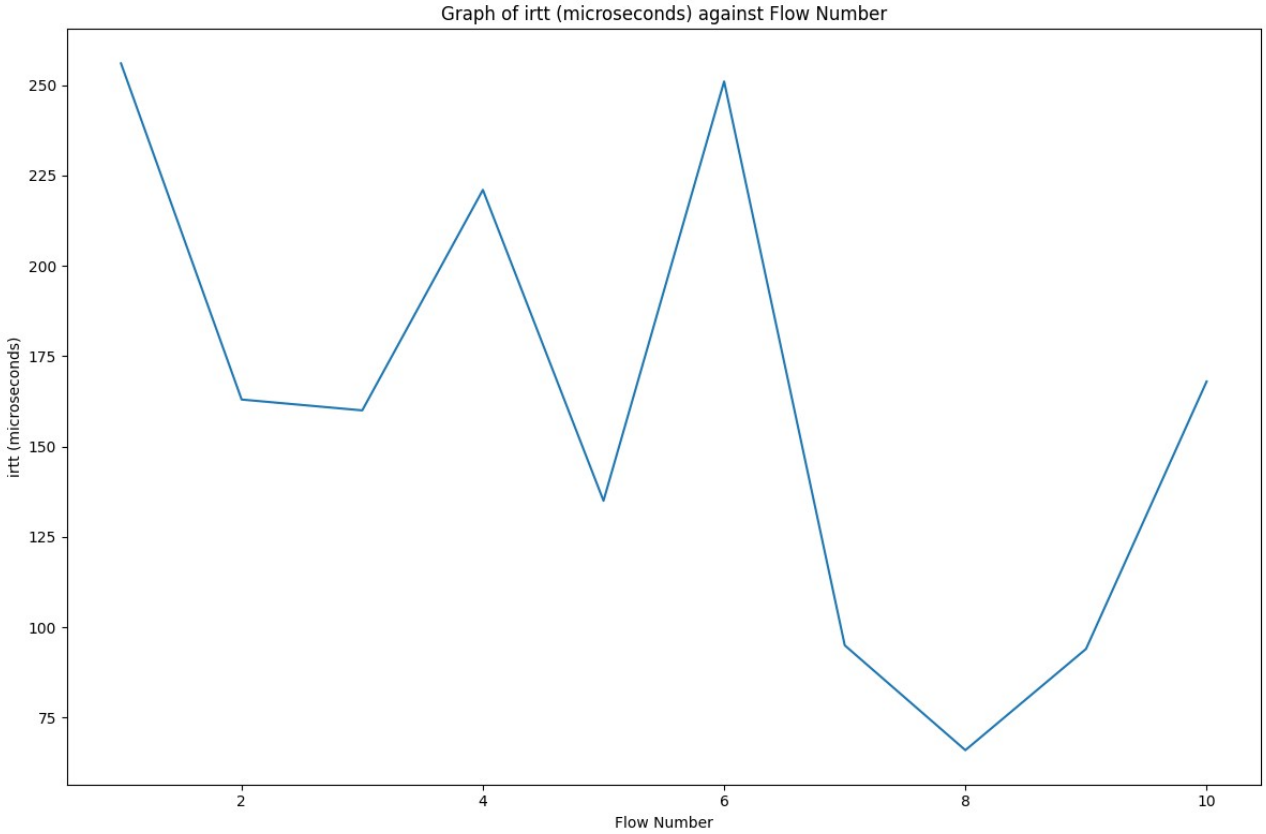
	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
count	10.0	10.0	10.0
mean	1.1	940.5	160.9
std	0.0	0.5270462766947299	66.10168933796876
min	1.1	940.0	66.0
25%	1.1	940.0	105.0
50%	1.1	940.5	161.5
75%	1.1	941.0	207.75
max	1.1	941.0	256.0

Graphs:



Module code: CS4105
Matriculation number: 200017941





Module code: CS4105
Matriculation number: 200017941

Without any delay, the mean bandwidth is 940.5 Mbits/sec with a standard deviation of 0.53. The data transferred is stationary at 1.1 Gbytes.

TCP flows with a constant delay

- Delay = 10ms

  Measurements table:

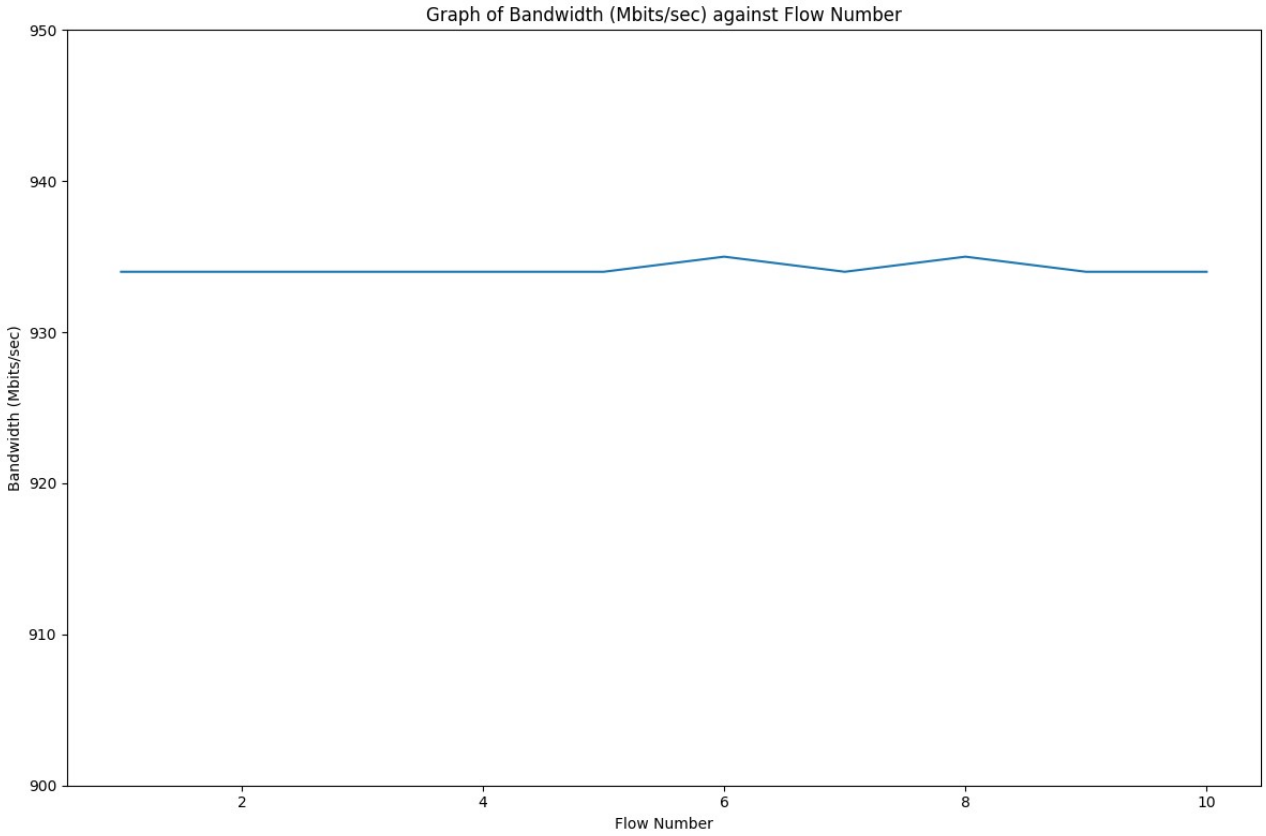
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	10.0	1.09	934.0	10308.0
1	10.0	1.09	934.0	10305.0
2	10.0	1.09	934.0	10320.0
3	10.0	1.09	934.0	10214.0
4	10.0	1.09	934.0	10241.0
5	10.0	1.09	935.0	10285.0
6	10.0	1.09	934.0	10260.0
7	10.0	1.09	935.0	10235.0
8	10.0	1.09	934.0	10286.0
9	10.0	1.09	934.0	10259.0

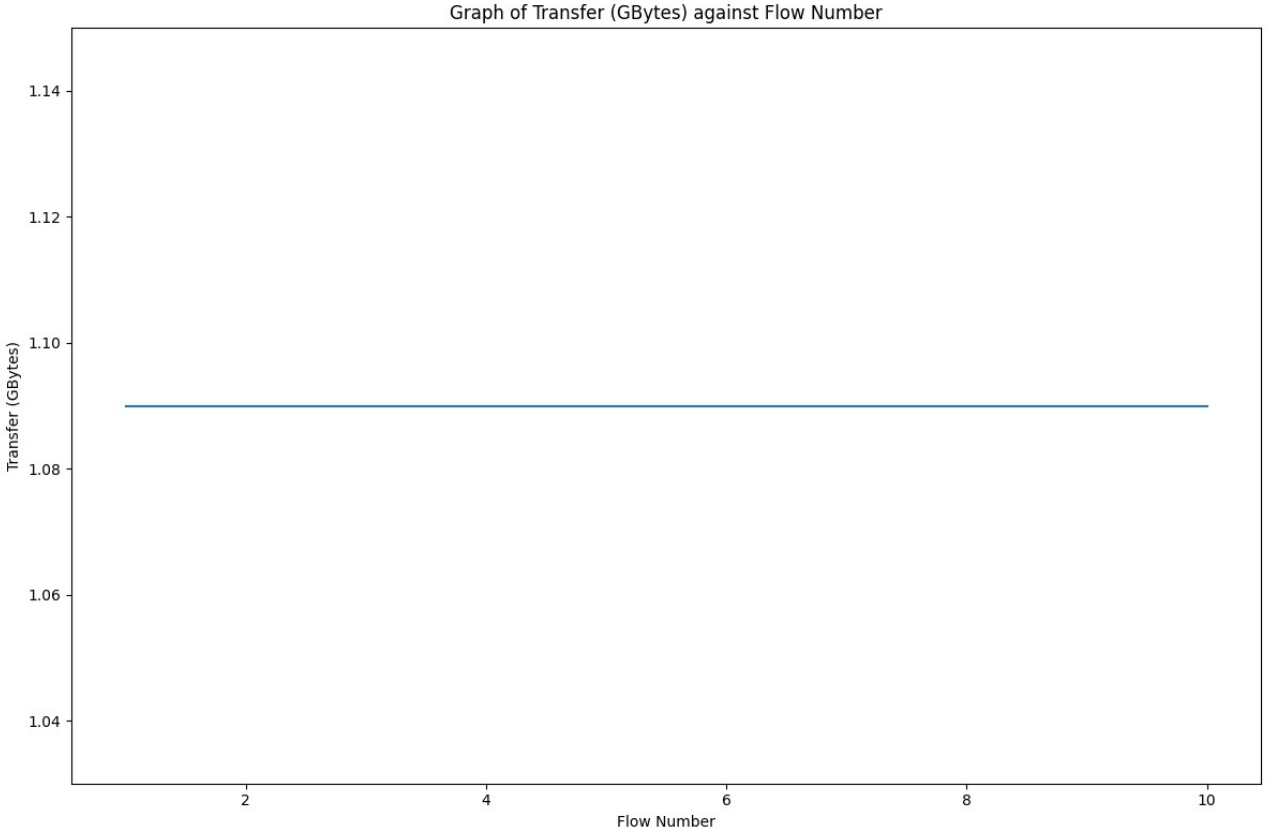
  Descriptive statistics:

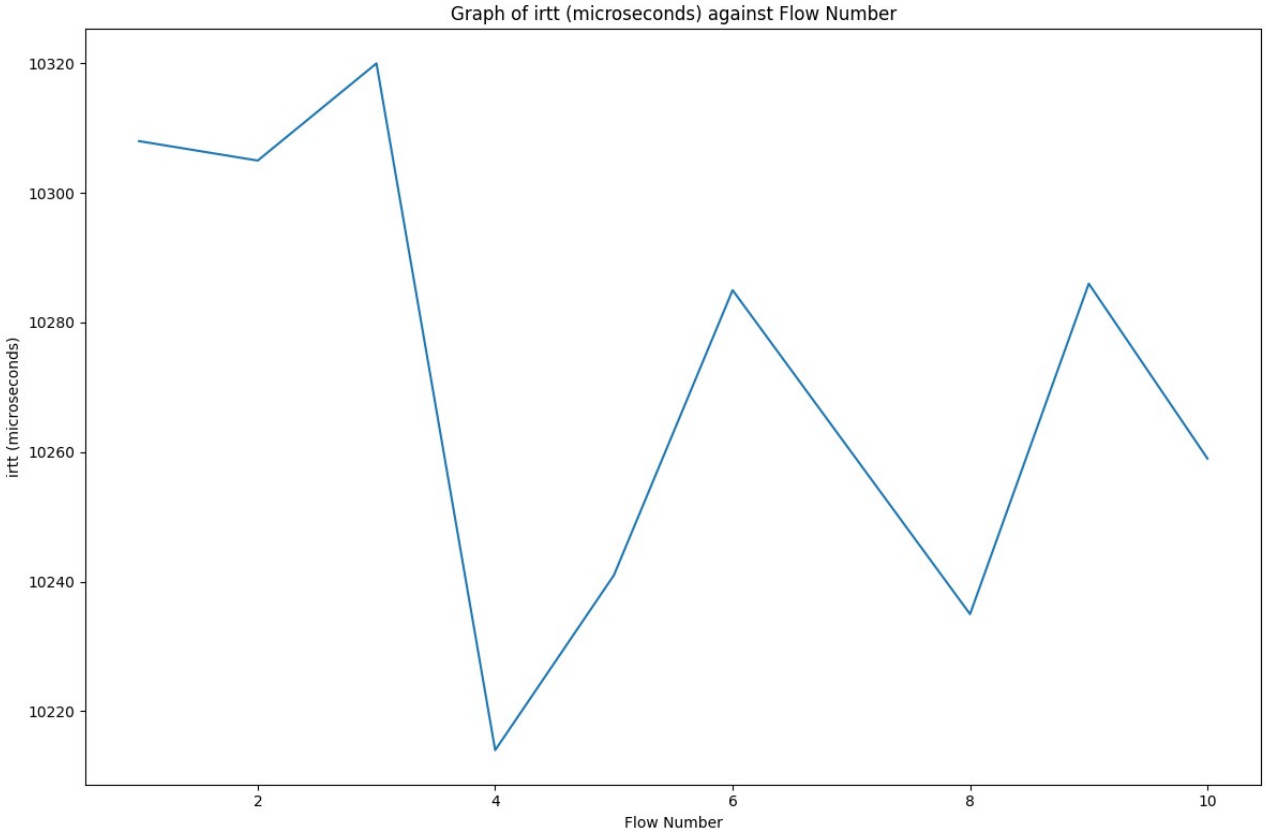
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	10.0	1.09	934.0	10308.0
1	10.0	1.09	934.0	10305.0
2	10.0	1.09	934.0	10320.0
3	10.0	1.09	934.0	10214.0
4	10.0	1.09	934.0	10241.0
5	10.0	1.09	935.0	10285.0
6	10.0	1.09	934.0	10260.0
7	10.0	1.09	935.0	10235.0
8	10.0	1.09	934.0	10286.0
9	10.0	1.09	934.0	10259.0

Module code: CS4105
Matriculation number: 200017941

Graphs:




Graph of Transfer (GBytes) against Flow Number
Transfer (GBytes)
1.14
1.12
1.10
1.08
1.06
1.04
2
4
6
8
10
Flow Number


Graph of irtt (microseconds) against Flow Number
irtt (microseconds)
10320
10300
10280
10260
10240
10220
2
4
6
8
10
Flow Number

When the delay is 10ms, the mean bandwidth is 934.2 Mbits/sec with a standard deviation of 0.42. The data transferred is stationary at 1.09 Gbytes.

- Delay = 20ms

Measurements table:

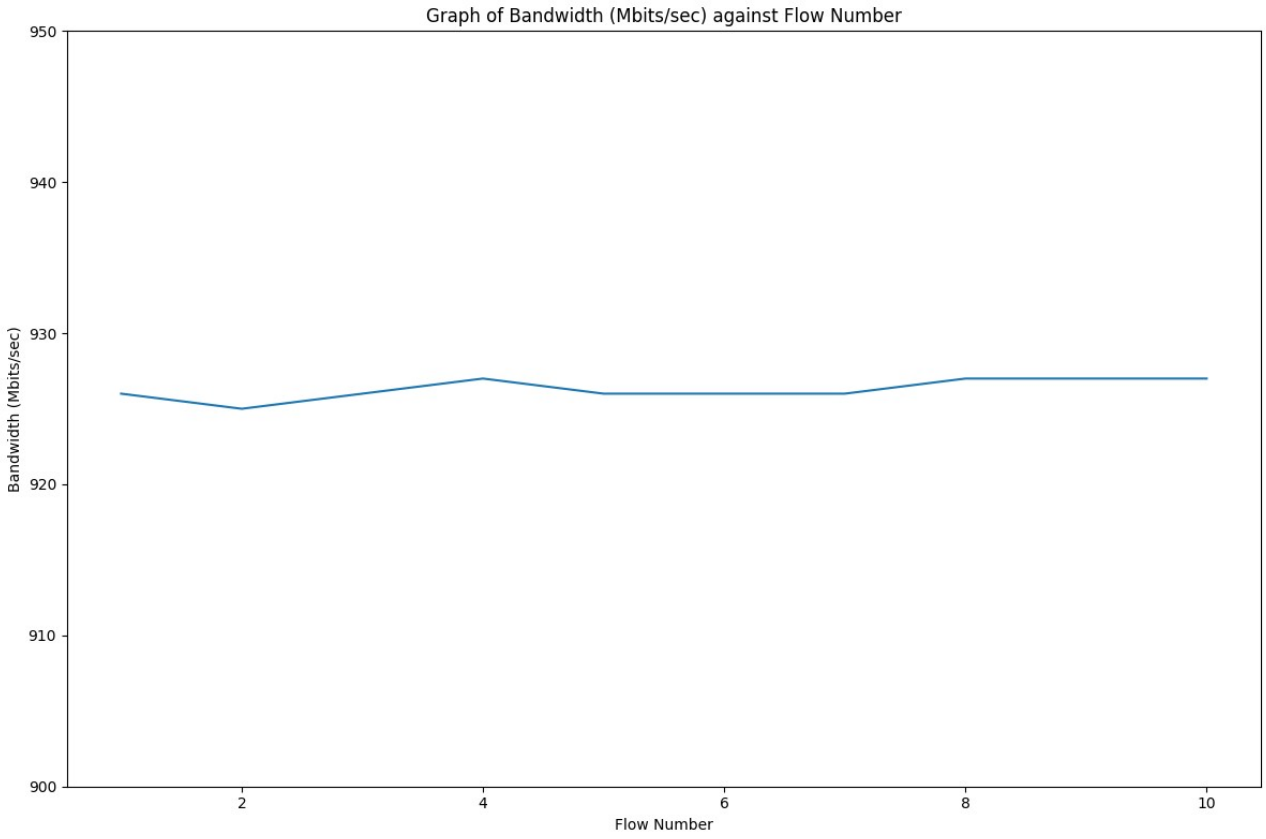
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	20.0	1.08	926.0	20276.0
1	20.0	1.08	925.0	20289.0
2	20.0	1.08	926.0	20289.0
3	20.0	1.08	927.0	20283.0
4	20.0	1.08	926.0	20285.0
5	20.0	1.08	926.0	20284.0
6	20.0	1.08	926.0	20286.0
7	20.0	1.08	927.0	20282.0
8	20.0	1.08	927.0	20287.0
9	20.0	1.08	927.0	20225.0

Descriptive statistics:

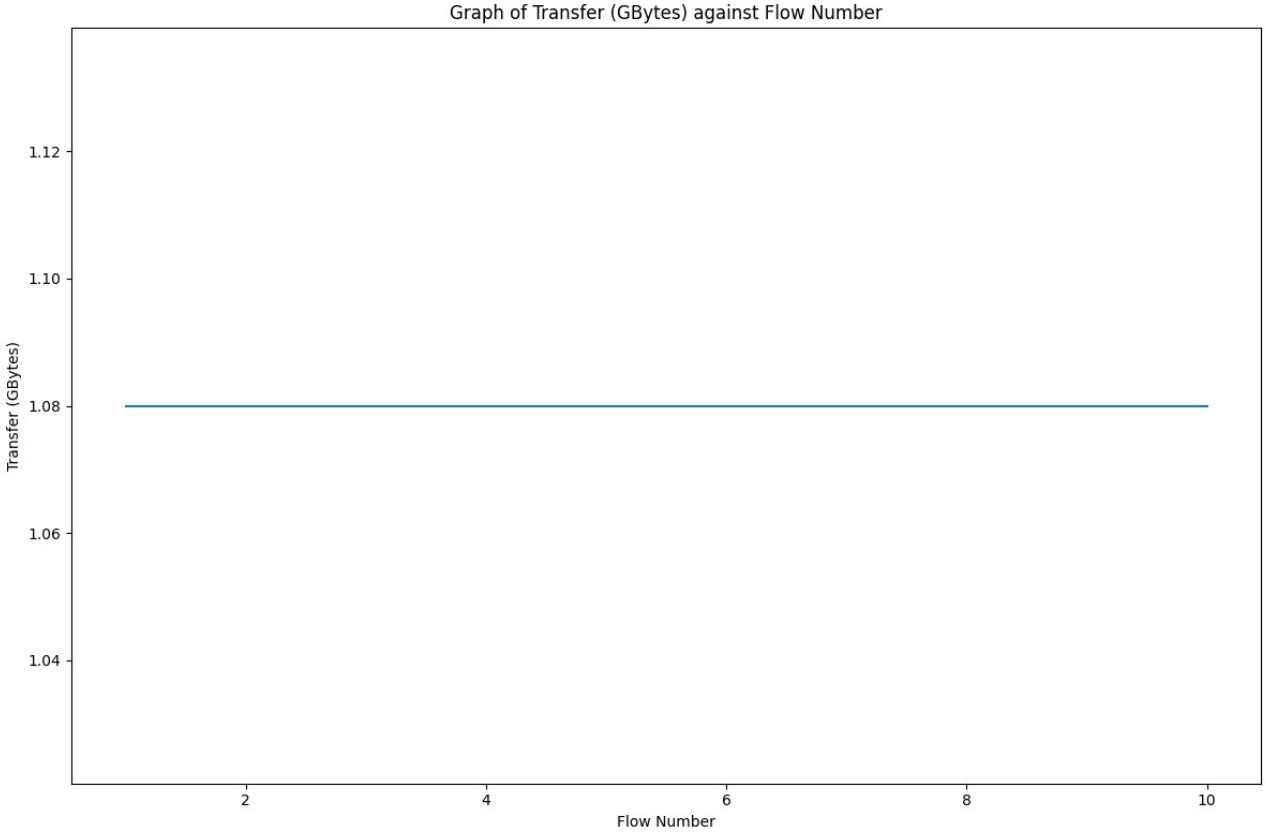
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	20.0	1.08	926.0	20276.0
1	20.0	1.08	925.0	20289.0
2	20.0	1.08	926.0	20289.0
3	20.0	1.08	927.0	20283.0
4	20.0	1.08	926.0	20285.0
5	20.0	1.08	926.0	20284.0
6	20.0	1.08	926.0	20286.0
7	20.0	1.08	927.0	20282.0
8	20.0	1.08	927.0	20287.0
9	20.0	1.08	927.0	20225.0

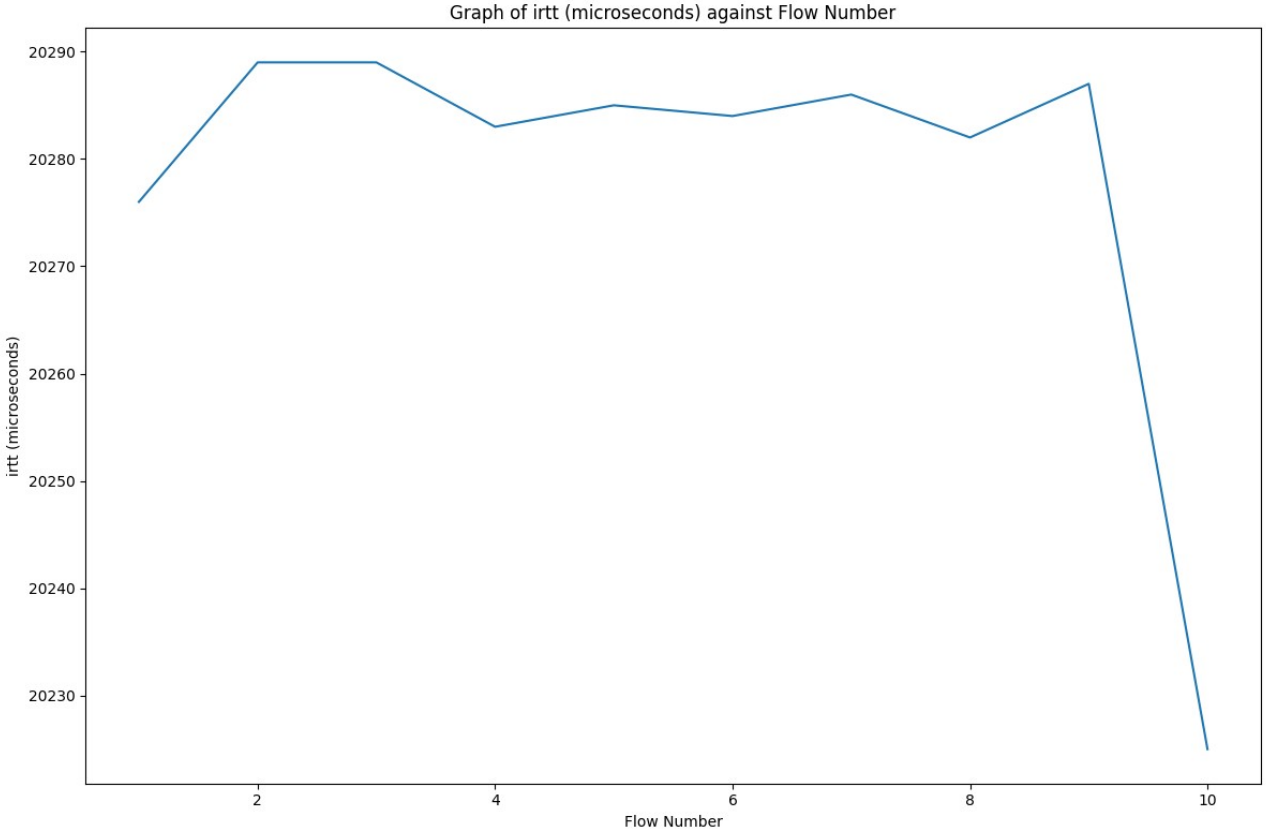
Module code: CS4105
Matriculation number: 200017941

Graphs:



Module code: CS4105
Matriculation number: 200017941





When the delay is 20ms, the mean bandwidth is 936.3 Mbits/sec with a standard deviation of 0.67. The data transferred is stationary at 1.08 Gbytes.

- Delay = 30ms

  Measurements table:

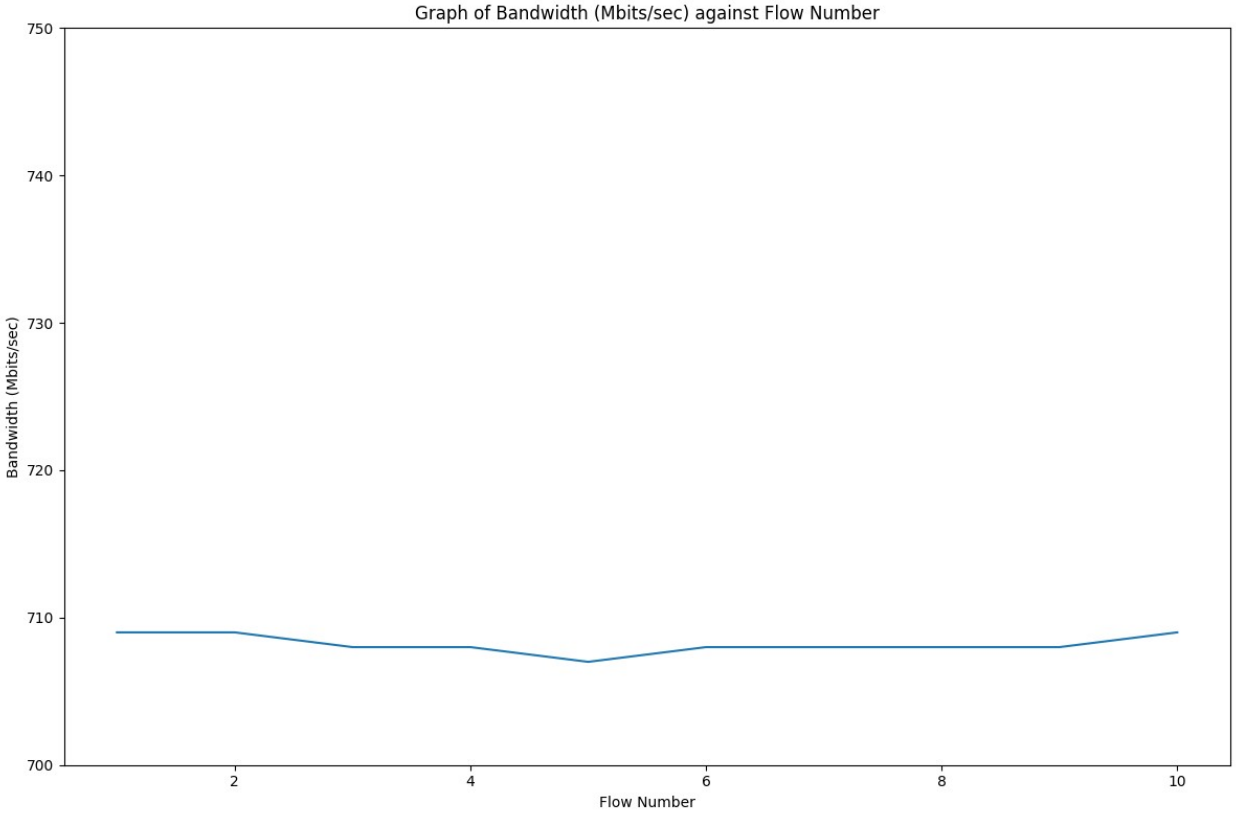
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	30.0	0.829	709.0	30261.0
1	30.0	0.829	709.0	30273.0
2	30.0	0.828	708.0	30263.0
3	30.0	0.828	708.0	30312.0
4	30.0	0.827	707.0	30263.0
5	30.0	0.828	708.0	30258.0
6	30.0	0.828	708.0	30258.0
7	30.0	0.828	708.0	30263.0
8	30.0	0.828	708.0	30299.0
9	30.0	0.829	709.0	30265.0

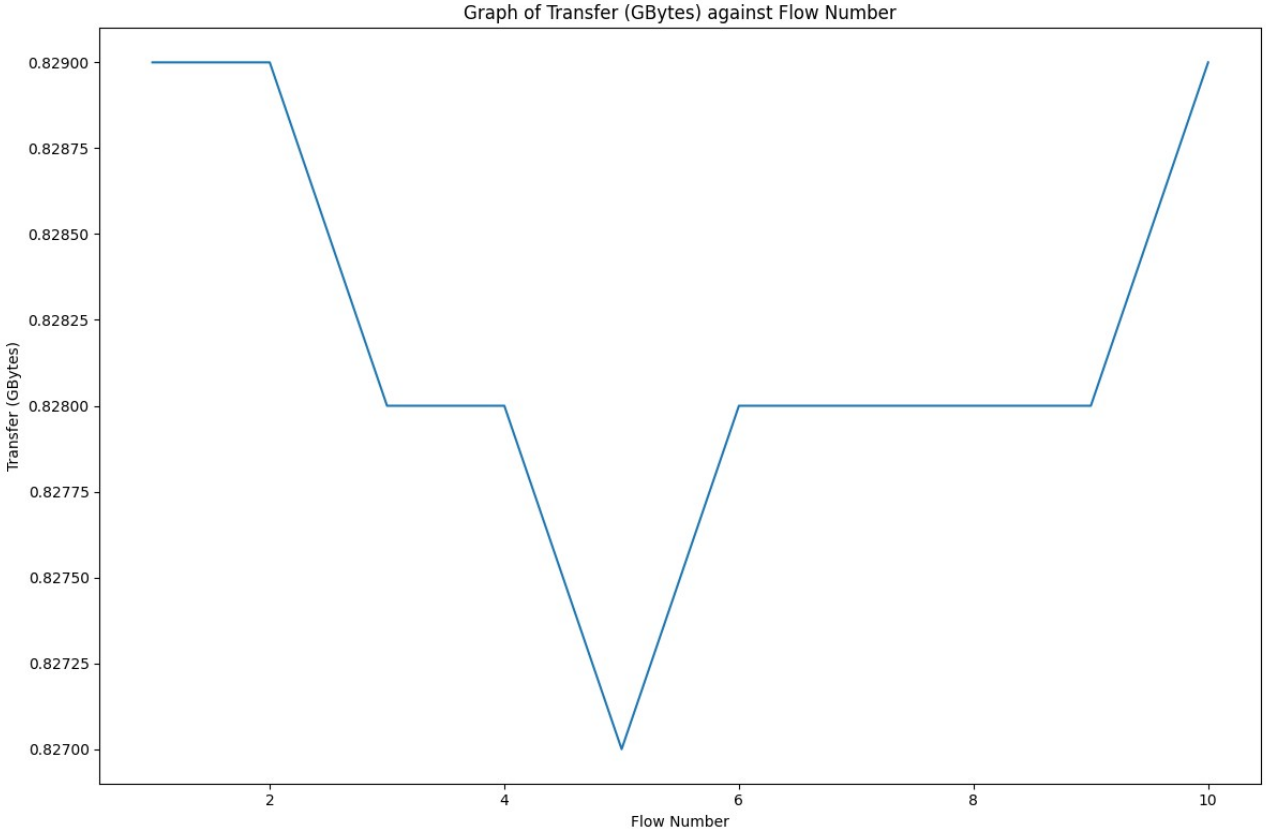
Module code: CS4105
Matriculation number: 200017941

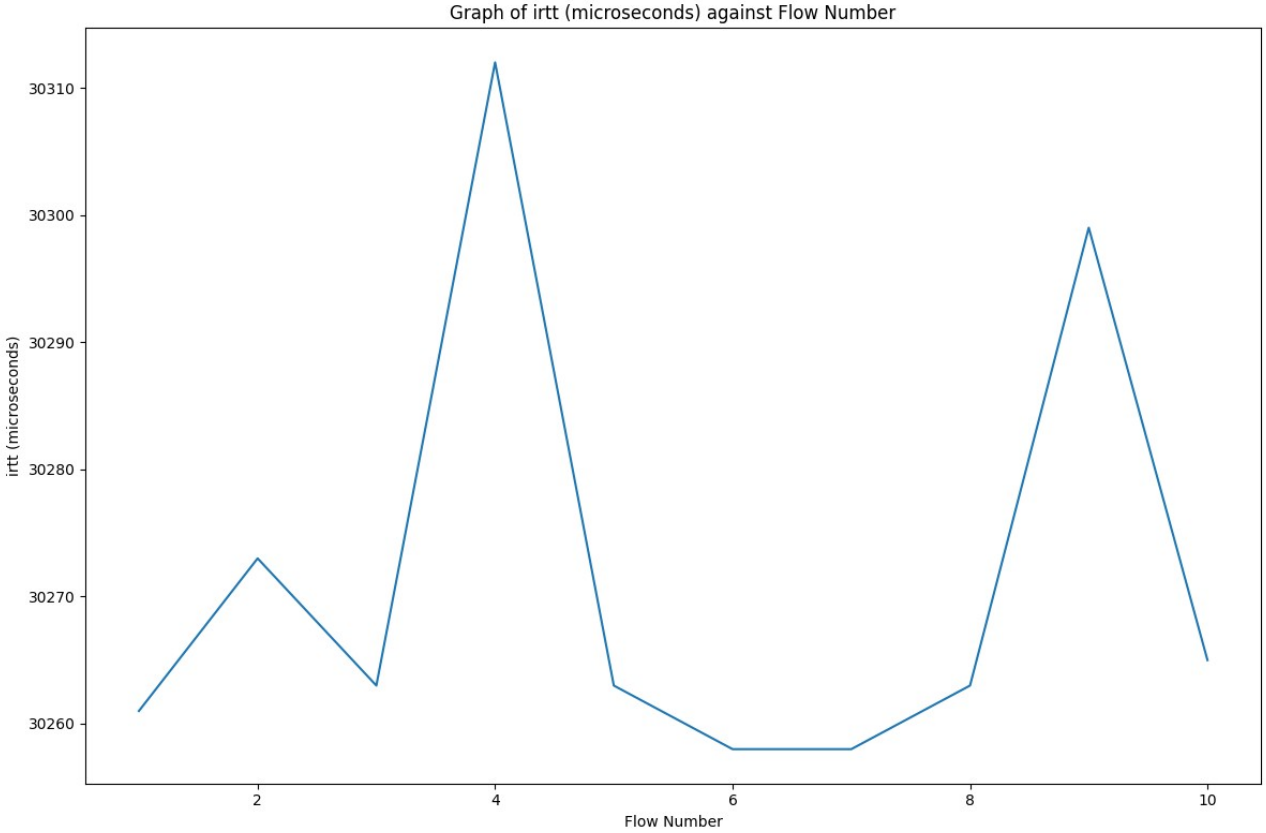
Descriptive statistics:

	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
count	10.0	10.0	10.0
mean	0.8282	708.2	30271.5
std	0.0006324555320336764	0.6324555320336759	18.656247091940962
min	0.827	707.0	30258.0
25%	0.828	708.0	30261.5
50%	0.828	708.0	30263.0
75%	0.82875	708.75	30271.0
max	0.829	709.0	30312.0

Graphs:




Graph of Transfer (GBytes) against Flow Number
Transfer (GBytes)
0.82900
0.82875
0.82850
0.82825
0.82800
0.82775
0.82750
0.82725
0.82700
2
4
6
8
10
Flow Number



When the delay is 30ms, the mean bandwidth is 708.2 Mbits/sec with a standard deviation of 0.63. The mean data transferred is 0.83 Gbytes with a standard deviation of 0.0006.

- Delay = 40ms

  Measurements table:

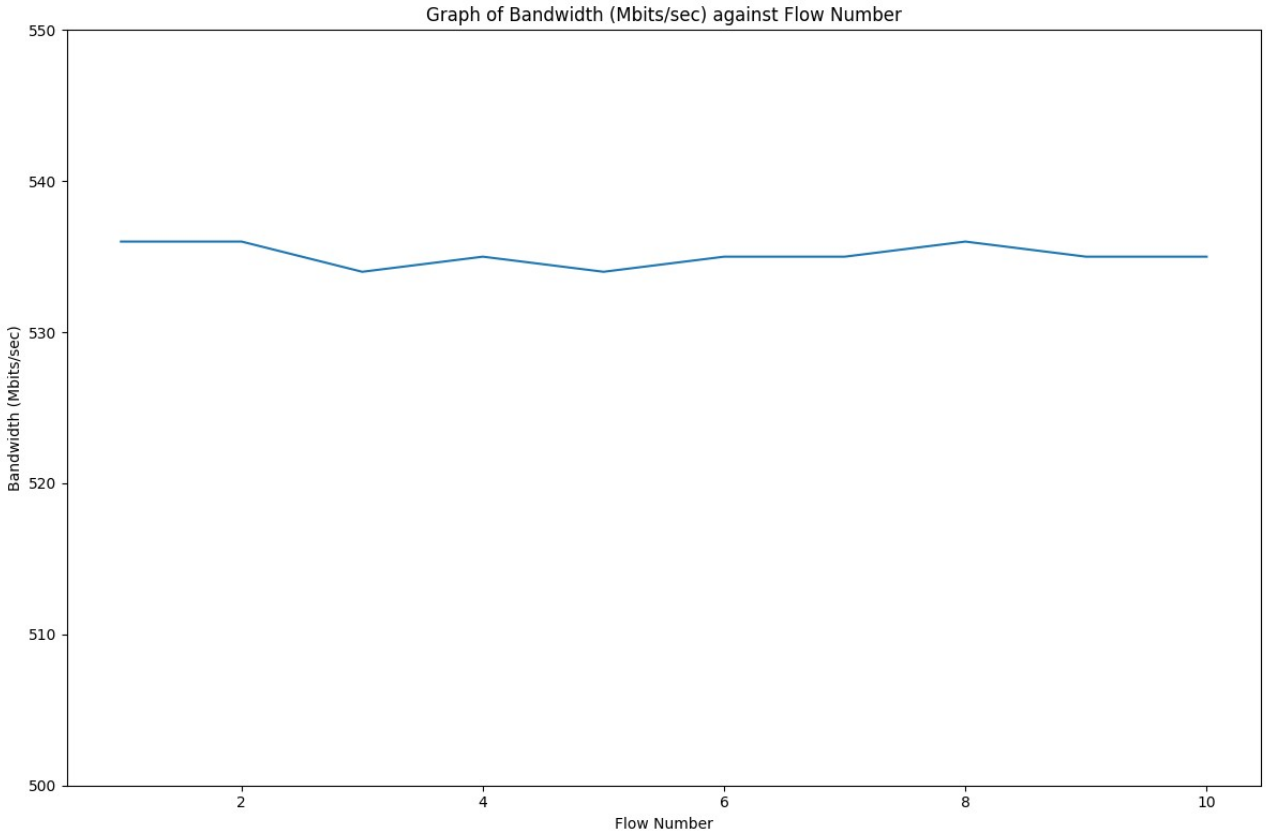
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	40.0	0.628	536.0	40267.0
1	40.0	0.628	536.0	40258.0
2	40.0	0.626	534.0	40242.0
3	40.0	0.627	535.0	40207.0
4	40.0	0.627	534.0	40261.0
5	40.0	0.627	535.0	40261.0
6	40.0	0.627	535.0	40265.0
7	40.0	0.627	536.0	40261.0
8	40.0	0.626	535.0	40228.0
9	40.0	0.626	535.0	40257.0

Module code: CS4105
Matriculation number: 200017941

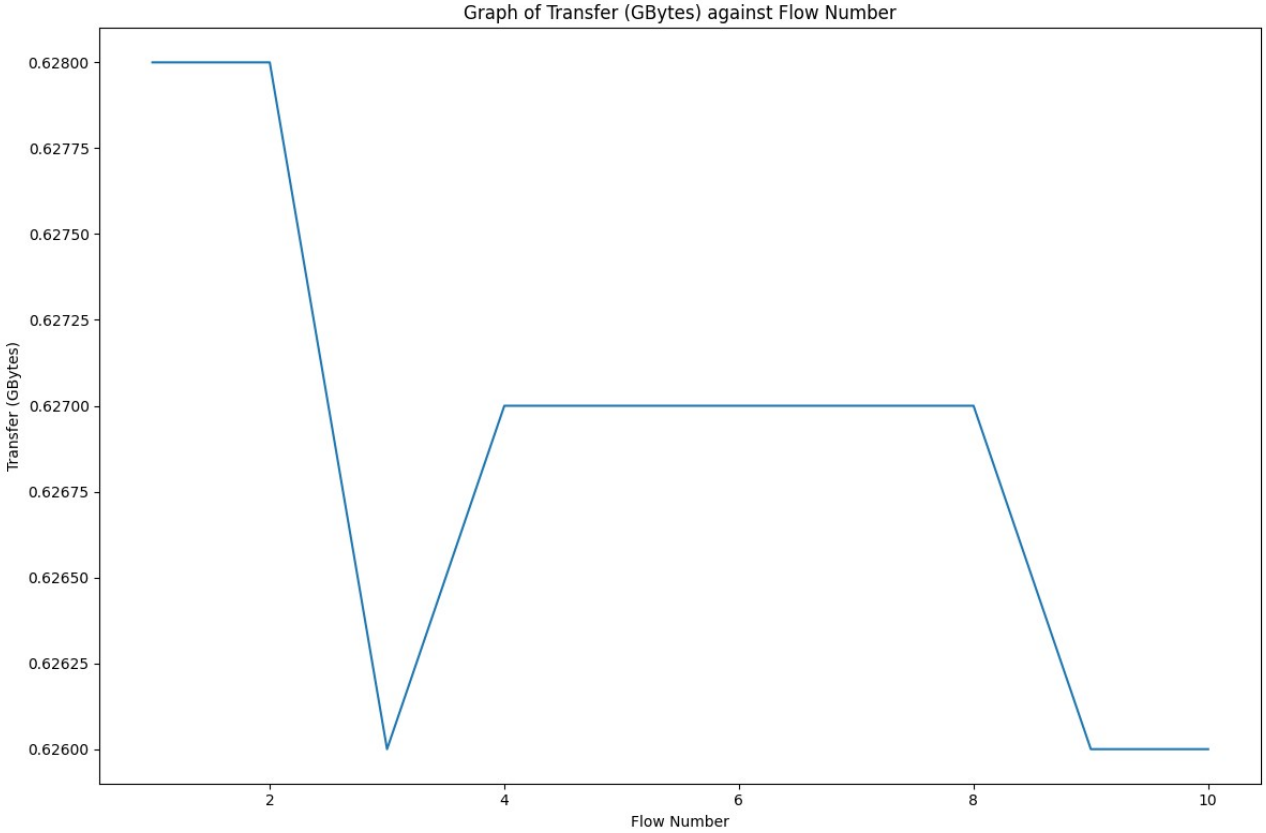
Descriptive statistics:

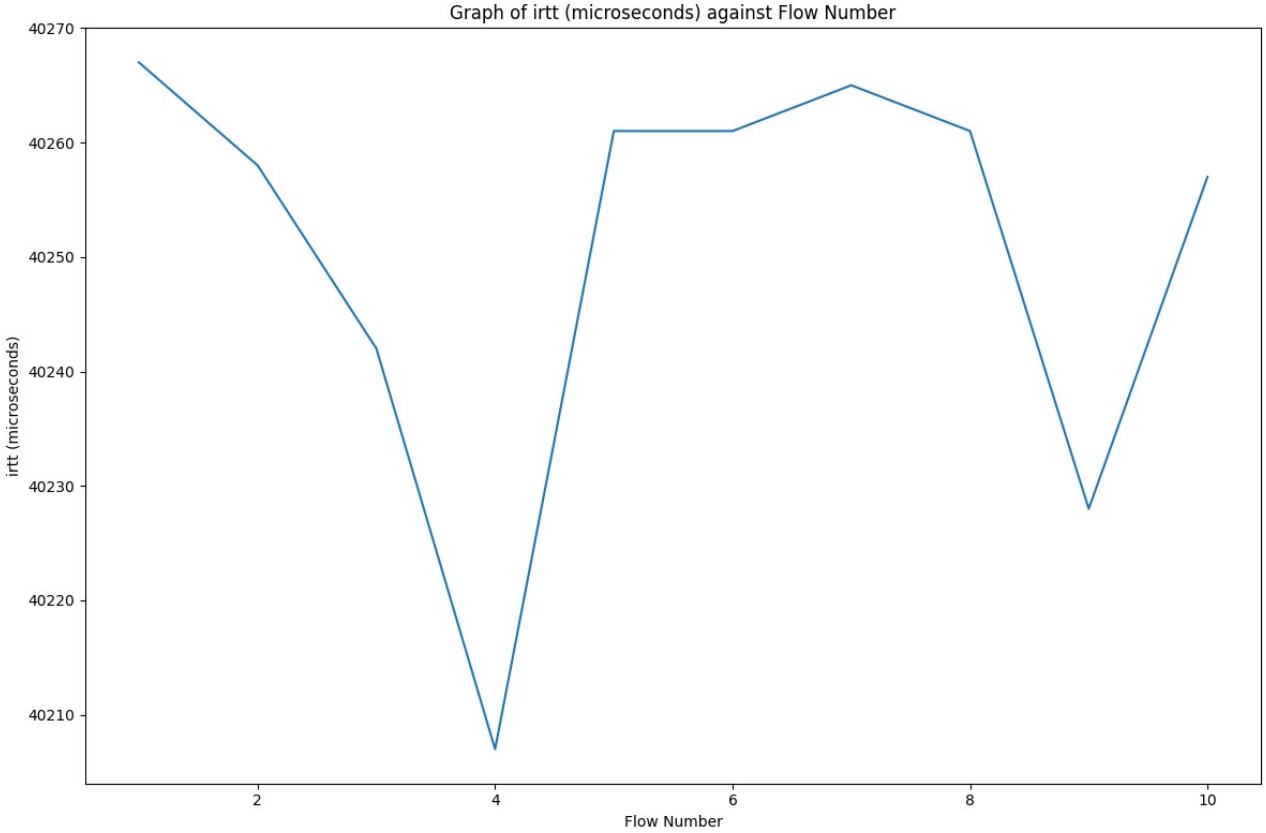
	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
count	10.0	10.0	10.0
mean	0.6269000000000001	535.1	40250.7
std	0.0007378647873726225	0.7378647873726217	19.3852750532173
min	0.626	534.0	40207.0
25%	0.62625	535.0	40245.75
50%	0.627	535.0	40259.5
75%	0.627	535.75	40261.0
max	0.628	536.0	40267.0

Graphs:



Module code: CS4105
Matriculation number: 200017941





When the delay is 40ms, the mean bandwidth is 535.1 Mbits/sec with a standard deviation of 0.74. The mean data transferred is 0.63 Gbytes with a standard deviation of 0.0007.

- Delay = 50ms

  Measurements table:

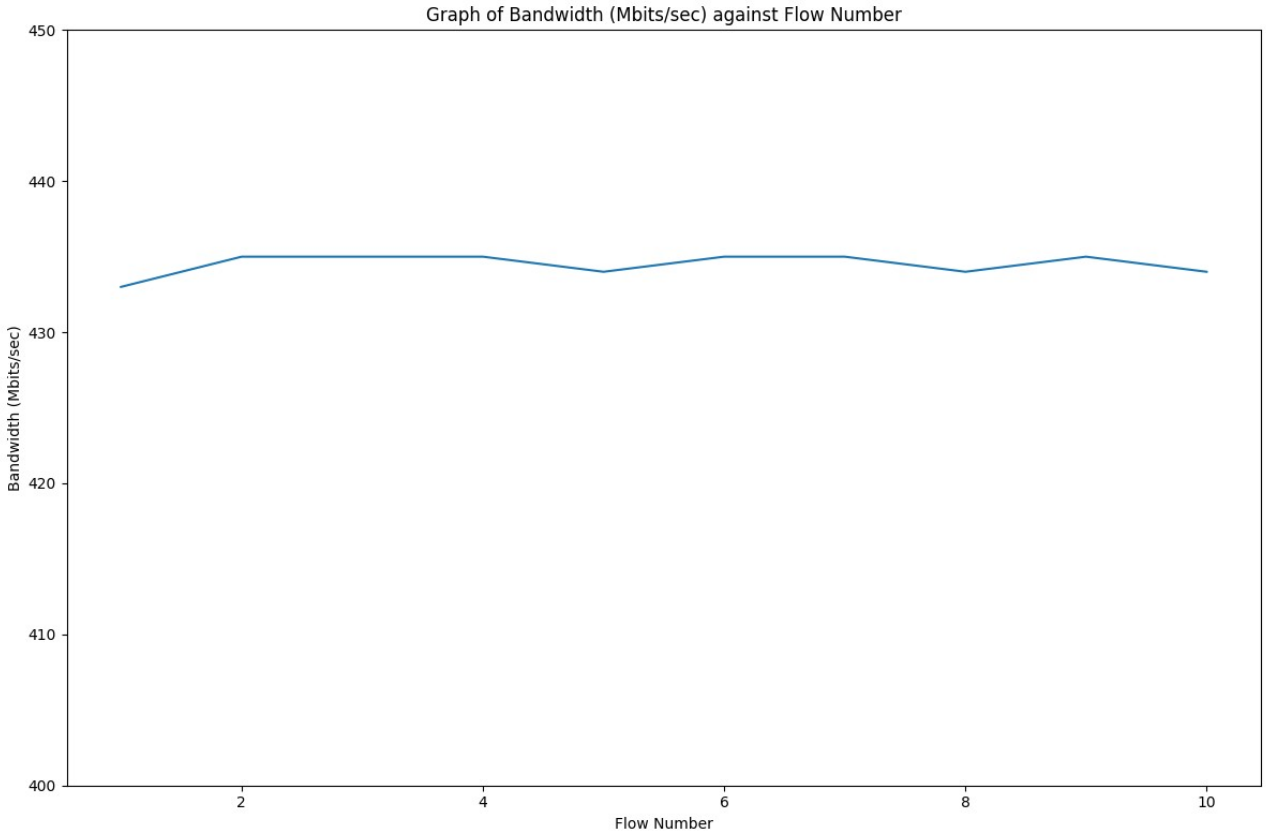
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	50.0	0.508	433.0	50264.0
1	50.0	0.51	435.0	50262.0
2	50.0	0.51	435.0	50225.0
3	50.0	0.511	435.0	50256.0
4	50.0	0.509	434.0	50260.0
5	50.0	0.511	435.0	50268.0
6	50.0	0.511	435.0	50262.0
7	50.0	0.509	434.0	50275.0
8	50.0	0.51	435.0	50273.0
9	50.0	0.509	434.0	50230.0

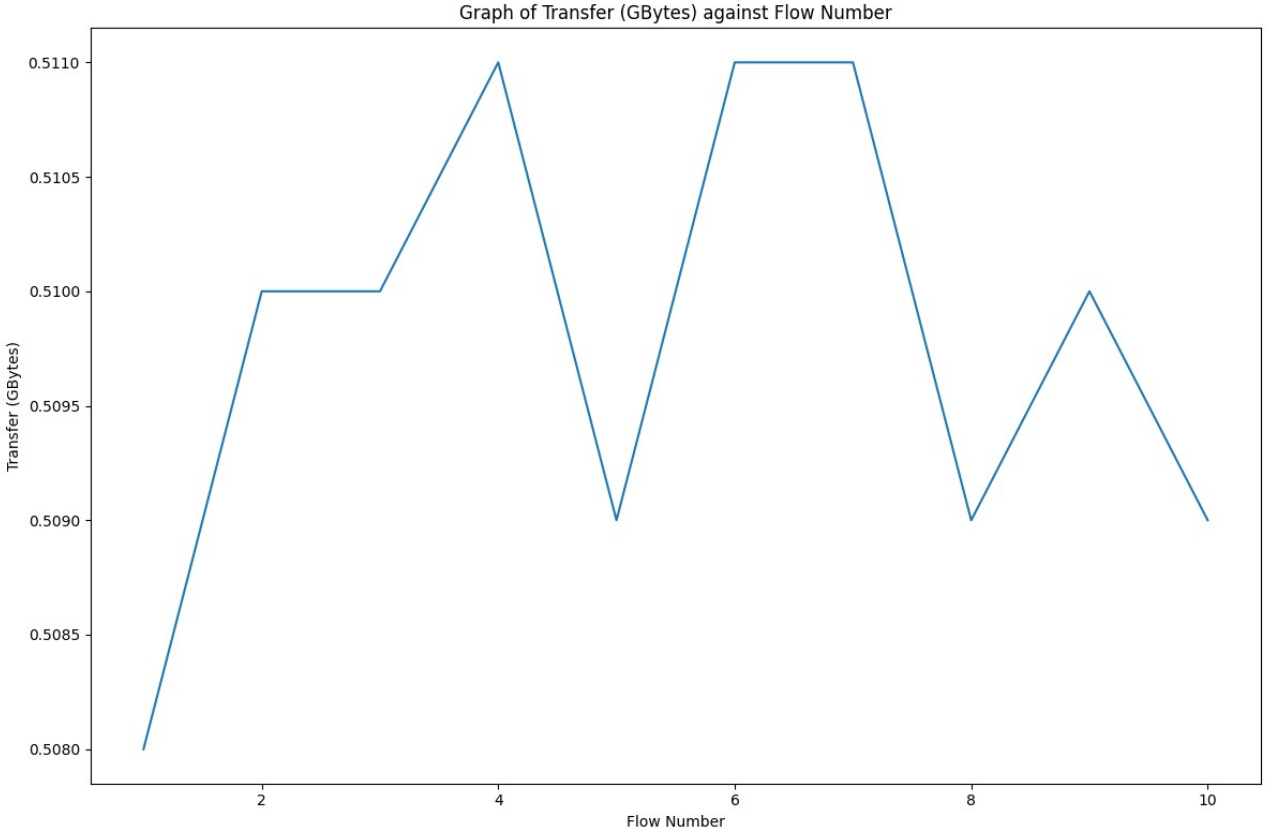
Module code: CS4105
Matriculation number: 200017941

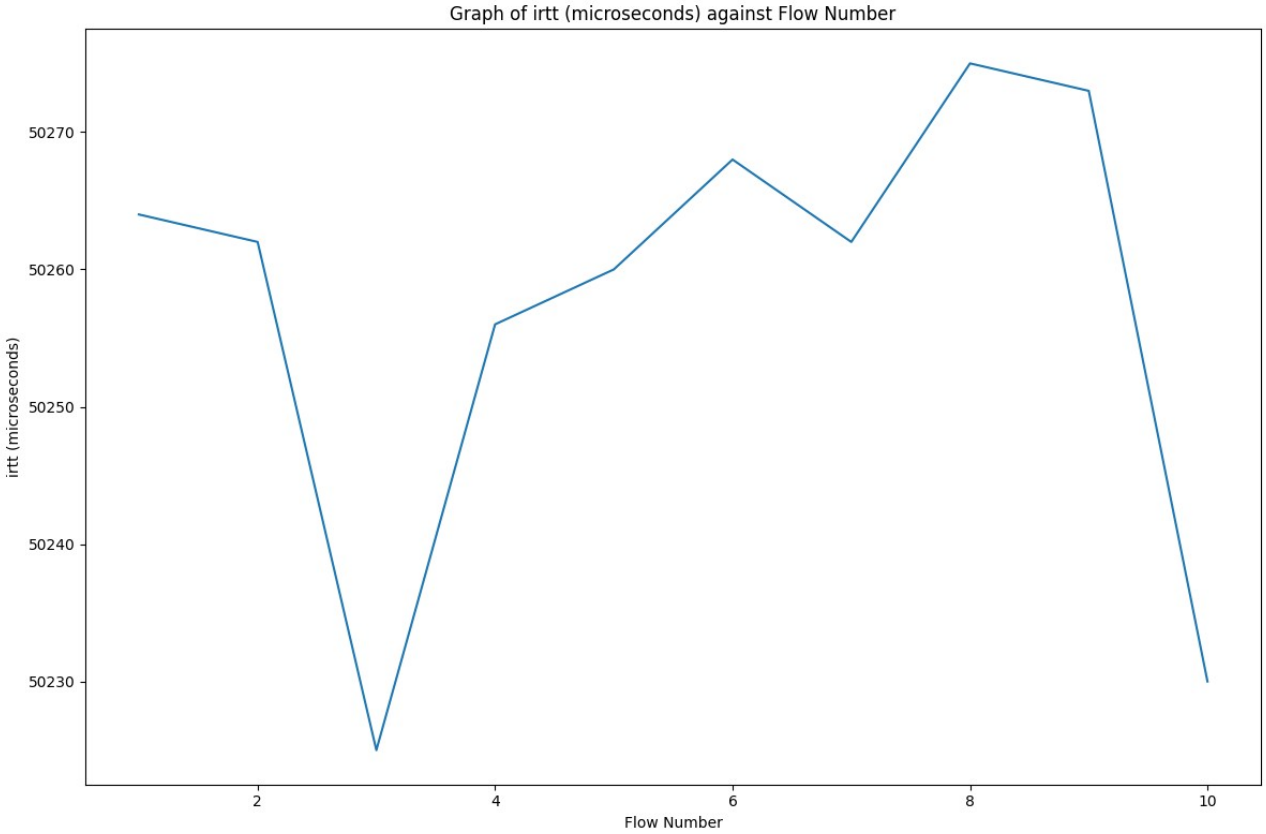
Descriptive statistics:

	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
count	10.0	10.0	10.0
mean	0.5098	434.5	50257.5
std	0.0010327955589886455	0.7071067811865476	16.867127793433
min	0.508	433.0	50225.0
25%	0.509	434.0	50257.0
50%	0.51	435.0	50262.0
75%	0.51075	435.0	50267.0
max	0.511	435.0	50275.0

Graphs:




Graph of Transfer (GBytes) against Flow Number
Transfer (GBytes)
0.5110
0.5105
0.5100
0.5095
0.5090
0.5085
0.5080
2
4
6
8
10
Flow Number


Graph of irtt (microseconds) against Flow Number
irtt (microseconds)
50270
50260
50250
50240
50230
2
4
6
8
10
Flow Number

When the delay is 50ms, the mean bandwidth is 434.5 Mbits/sec with a standard deviation of 0.71. The mean data transferred is 0.51 Gbytes with a standard deviation of 0.001.

- Delay = 60ms

Measurements table:

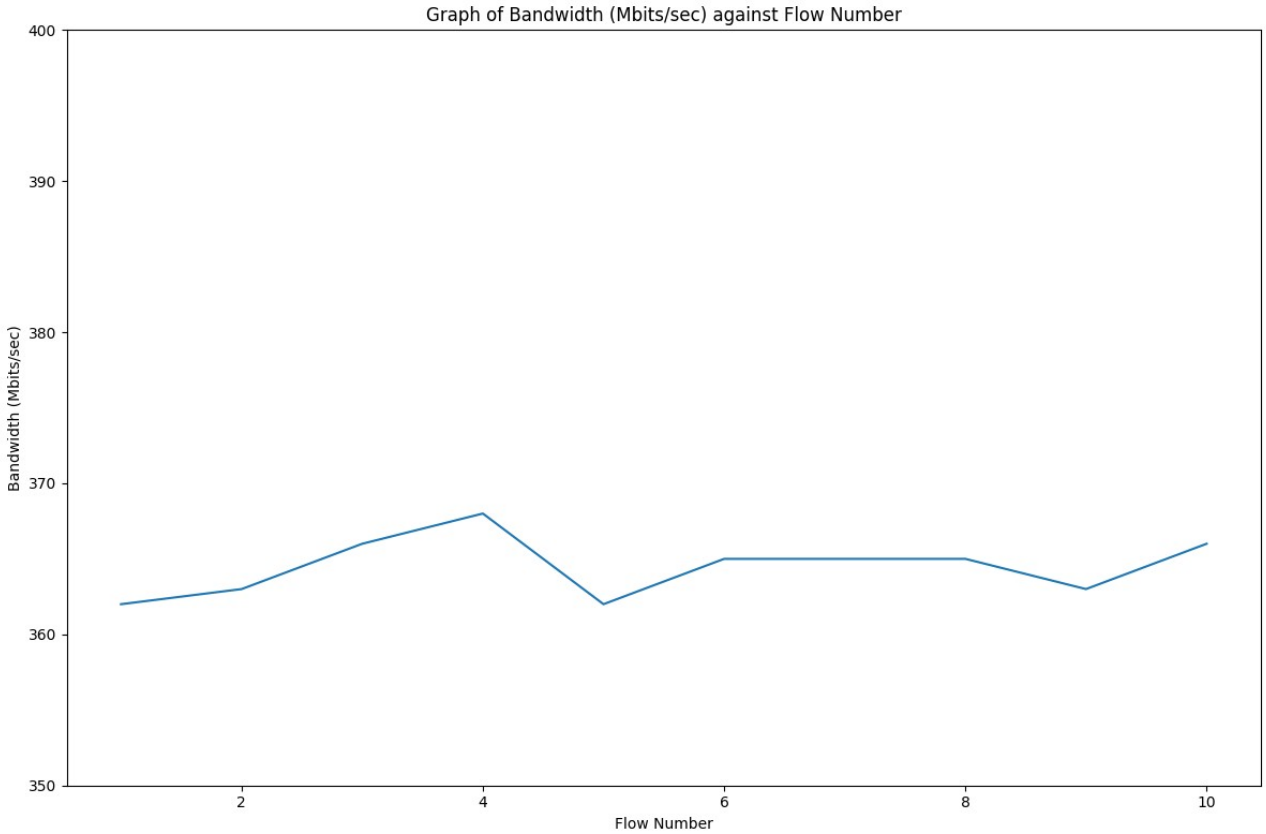
	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	60.0	0.426	362.0	60282.0
1	60.0	0.427	363.0	60231.0
2	60.0	0.43	366.0	60266.0
3	60.0	0.432	368.0	60265.0
4	60.0	0.425	362.0	60210.0
5	60.0	0.429	365.0	60210.0
6	60.0	0.428	365.0	60270.0
7	60.0	0.429	365.0	60260.0
8	60.0	0.427	363.0	60227.0
9	60.0	0.43	366.0	60264.0

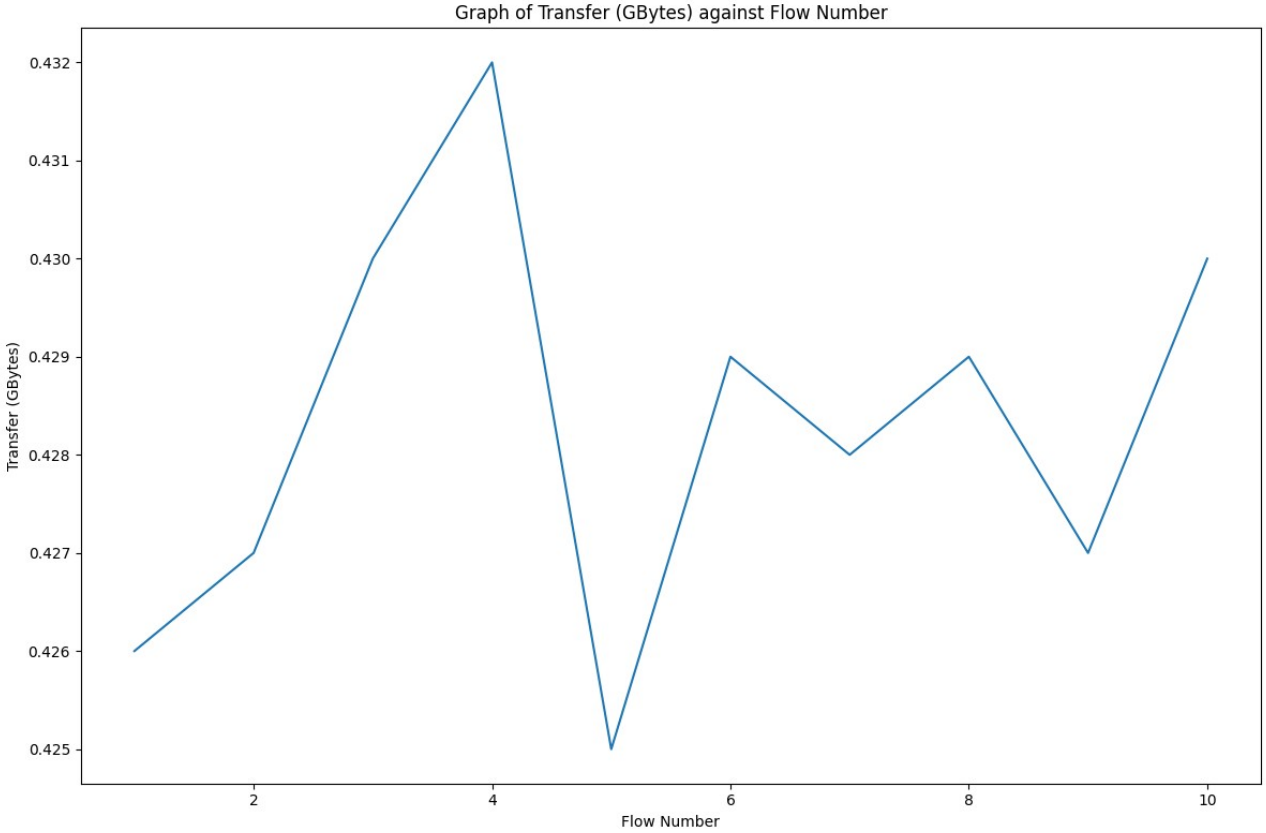
Descriptive statistics:

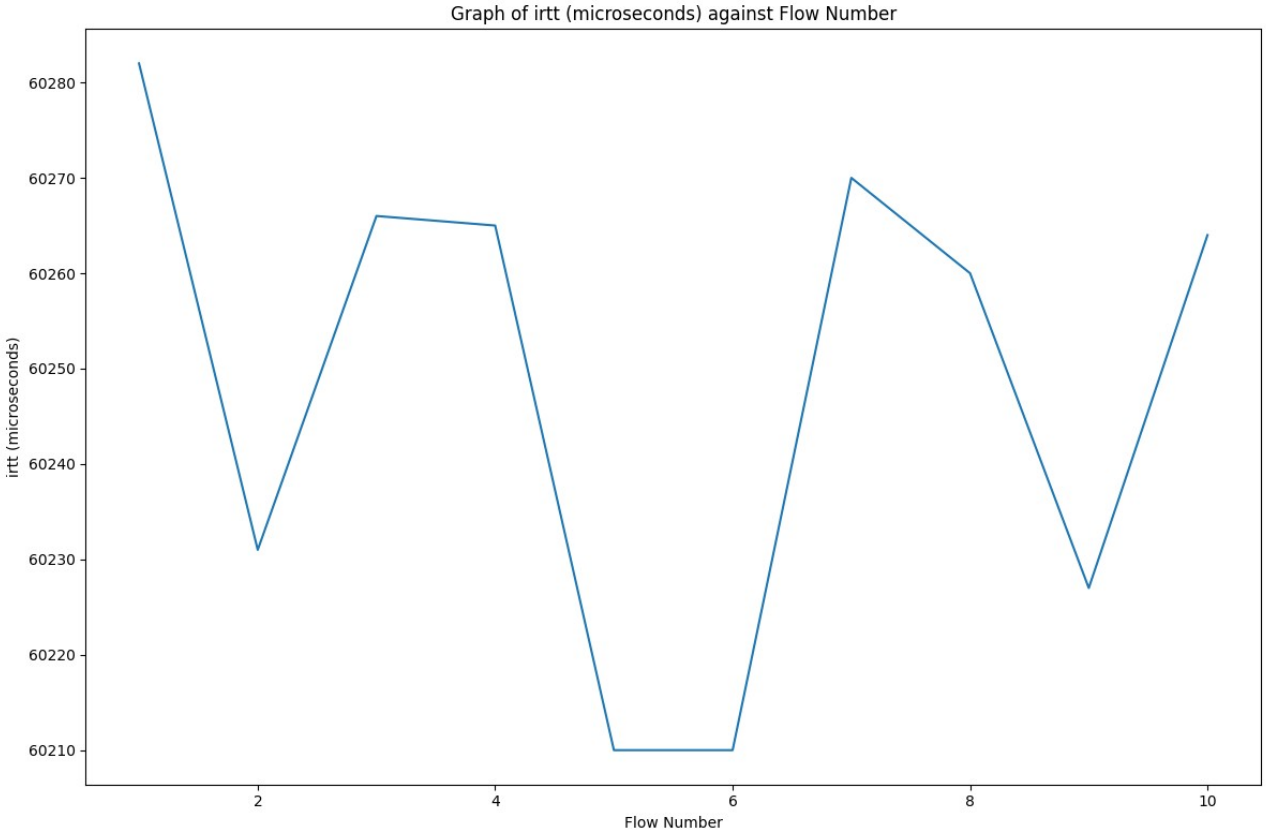
	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
count	10.0	10.0	10.0
mean	0.42829999999999996	364.5	60248.5
std	0.0021108186931983438	1.9578900207451218	26.391286440793294
min	0.425	362.0	60210.0
25%	0.427	363.0	60228.0
50%	0.4285	365.0	60262.0
75%	0.42974999999999997	365.75	60265.75
max	0.432	368.0	60282.0

Module code: CS4105
Matriculation number: 200017941

Graphs:



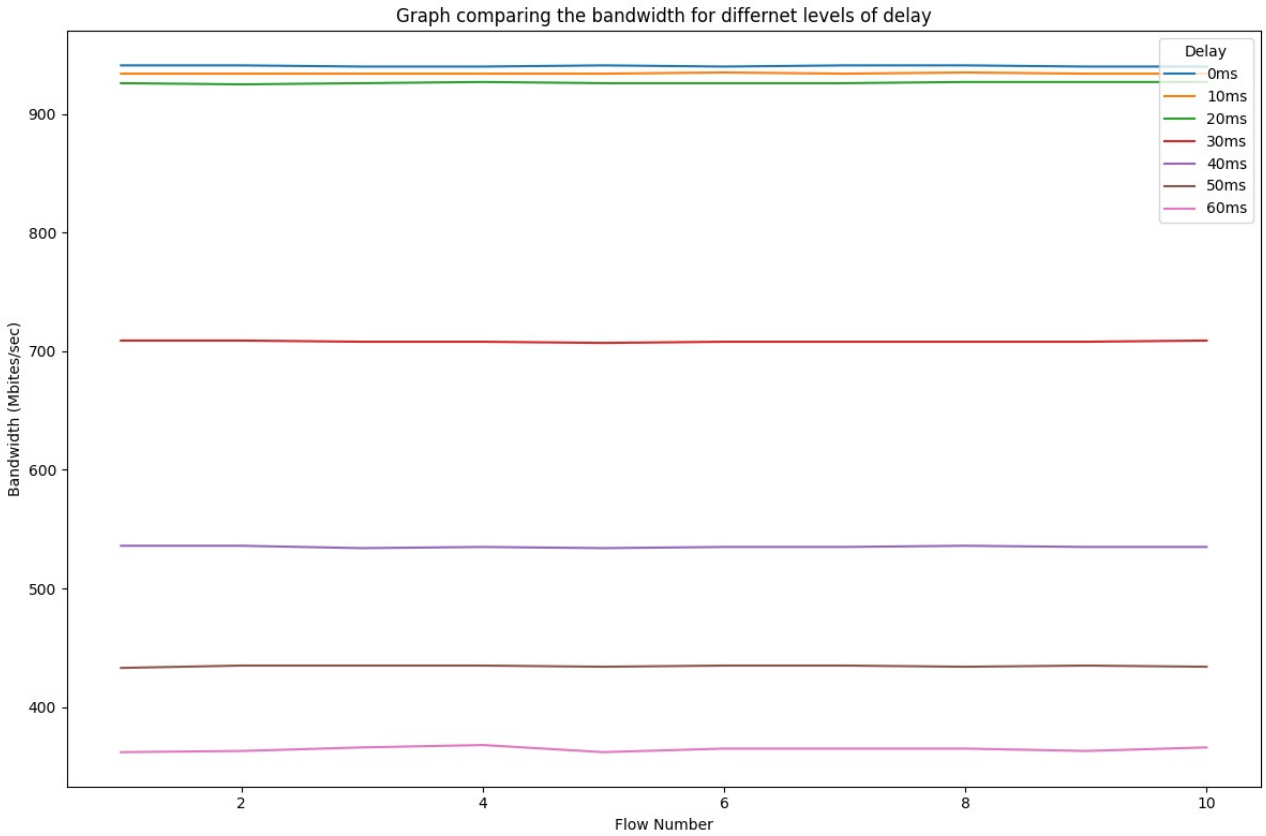

Graph of Transfer (GBytes) against Flow Number
Transfer (GBytes)
0.432
0.431
0.430
0.429
0.428
0.427
0.426
0.425
2
4
6
8
10
Flow Number


Graph of irtt (microseconds) against Flow Number
irtt (microseconds)
60280
60270
60260
60250
60240
60230
60220
60210
2
4
6
8
10
Flow Number

Module code: CS4105
Matriculation number: 200017941

When the delay is 60ms, the mean bandwidth is 364.5 Mbits/sec with a standard deviation of 1.96. The mean data transferred is 0.43 Gbytes with a standard deviation of 0.002.

Graph comparing the bandwidth across different levels of delay from the measurements above:



When the delay increases from 0ms to 20ms, there is only a slight drop in the average bandwidth. But from 20ms to 60ms, the average bandwidth decreases significantly, where the amount of decrease becomes smaller the higher the level of delay.

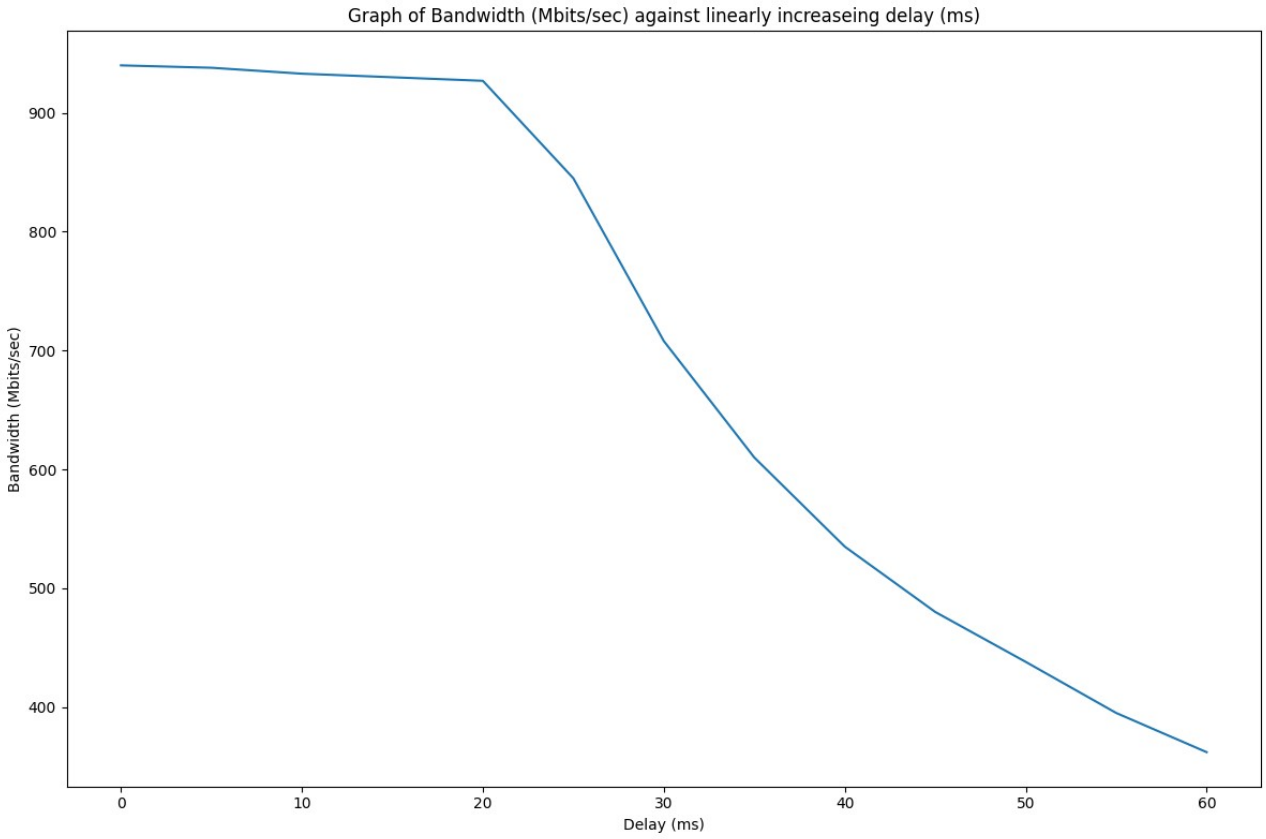
Module code: CS4105
Matriculation number: 200017941

TCP flows with a linear increase in delay

Measurements table:

	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	0.0	1.1	940.0	241.0
1	5.0	1.1	938.0	5285.0
2	10.0	1.09	933.0	10318.0
3	15.0	1.09	930.0	15297.0
4	20.0	1.08	927.0	20284.0
5	25.0	0.987	845.0	25232.0
6	30.0	0.829	708.0	30313.0
7	35.0	0.713	610.0	35262.0
8	40.0	0.627	535.0	40264.0
9	45.0	0.563	480.0	45223.0
10	50.0	0.513	438.0	50314.0
11	55.0	0.465	395.0	55256.0
12	60.0	0.426	362.0	60262.0

Graph:



When the delay is linearly increased (5ms), the bandwidth drops from 940.0 Mbits/sec to 362.0 Mbits/sec. The decrease in bandwidth up until a delay of 20ms is minimal, but the decrease afterwards is significant, which happens quadratically.

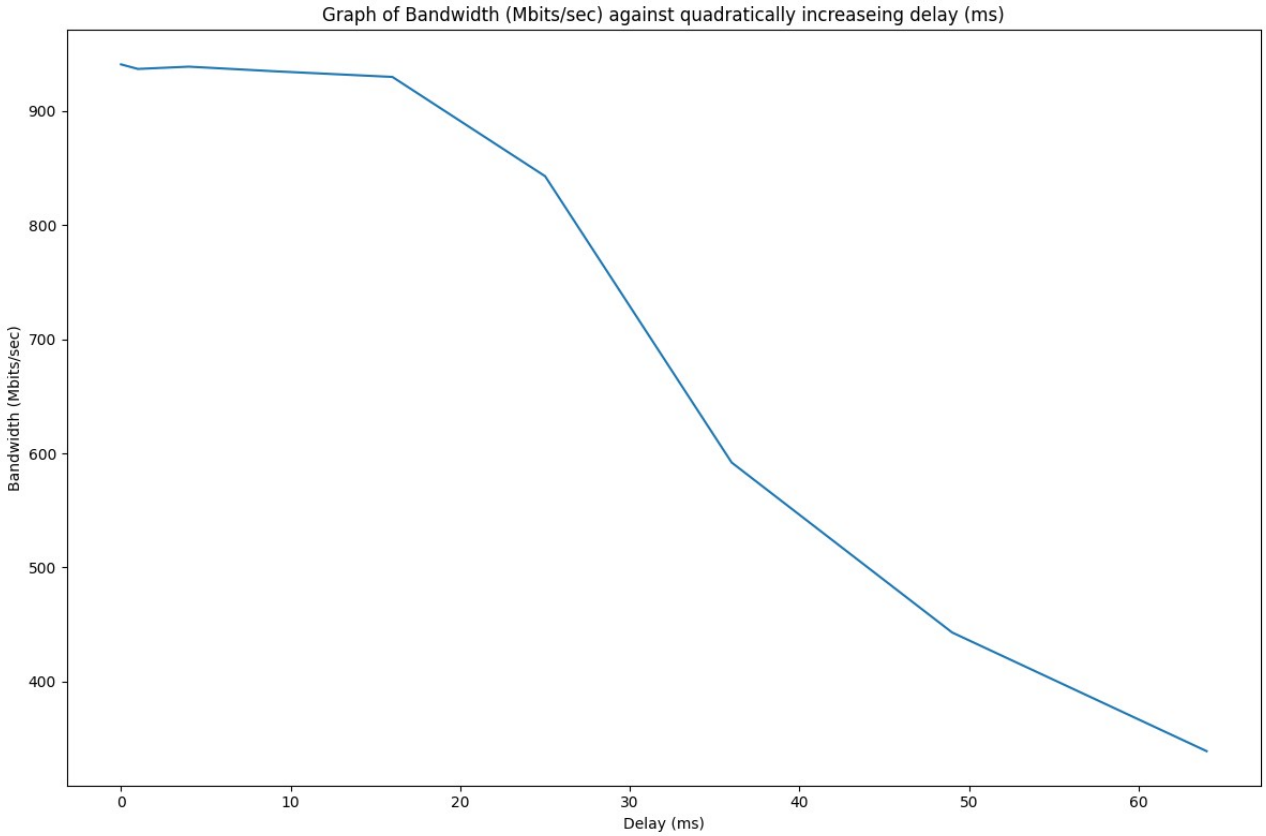
Module code: CS4105
Matriculation number: 200017941

TCP flows with a quadratic increase in delay

Measurements table:

	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	0.0	1.1	941.0	239.0
1	1.0	1.09	937.0	1204.0
2	4.0	1.1	939.0	4313.0
3	9.0	1.09	935.0	9270.0
4	16.0	1.09	930.0	16233.0
5	25.0	0.985	843.0	25324.0
6	36.0	0.693	592.0	36260.0
7	49.0	0.519	443.0	49232.0
8	64.0	0.398	339.0	64260.0

Graph:



When the delay is quadratically increased (x^2), the bandwidth drops from 941.0 Mbits/sec to 340.0 Mbits/sec. The decrease in bandwidth is first minimal up until when the delay is 16ms, but the decrease becomes significant afterwards, which happens quadratically.

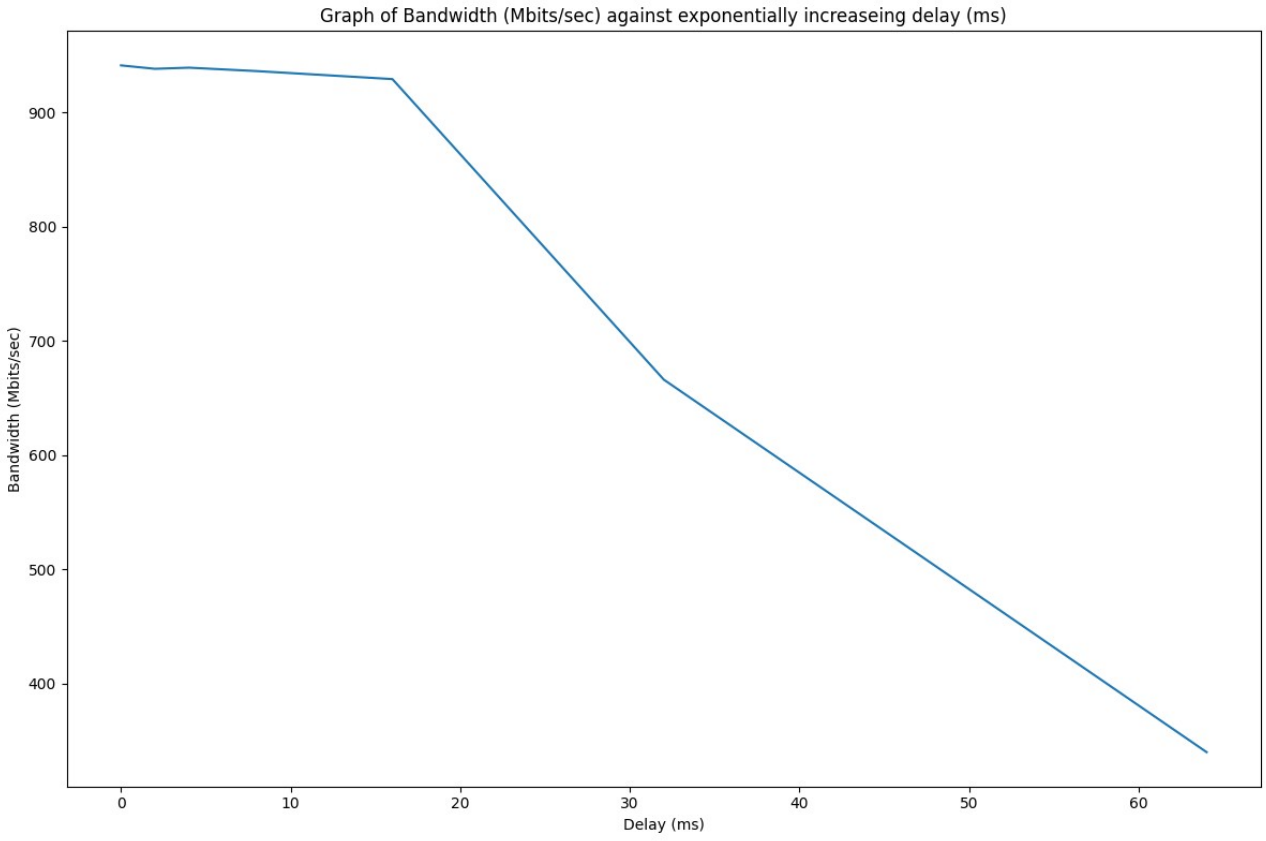
Module code: CS4105
Matriculation number: 200017941

TCP flows with an exponential increase in delay

Measurements table:

	Delay (ms)	Transfer (GBytes)	Bandwidth (Mbits/sec)	irtt (microseconds)
0	0.0	1.1	941.0	256.0
1	2.0	1.1	938.0	2348.0
2	4.0	1.1	939.0	4344.0
3	8.0	1.09	936.0	8304.0
4	16.0	1.09	929.0	16262.0
5	32.0	0.778	666.0	32254.0
6	64.0	0.398	340.0	64242.0

Graph:



When the delay is exponentially decreased (2^x), the bandwidth drops from 941.0 Mbits/sec to 340.0 Mbits/sec. The drop in bandwidth at first is minimal up until a delay of 8ms, but afterwards the drop is significant, but this happens relatively linearly.

Analyses and Discussion

The first main observation made from the collected measurements is that a decrease in bandwidth as the delay increase. This is what we would expect since throughput is measured as the amount of data transferred within a given time, which in this case is megabits per second. So, when the delay is increased, the time taken for the data to be transferred from the client to the server increases (time being in the denominator of the unit for throughput). This is due to the fact that TCP requires an acknowledgement to be sent back after data has been received by the server. Since delays are being added, TCP has to wait longer for the acknowledgement to be sent and received, which means that the available bandwidth is not being fully utilised and so is lower than 950 megabits per second which is the default bandwidth with no delays. The second observation is that the amount of data transferred delay decreases as the delay decreases which is the immediate result following a decrease in the bandwidth. This is because a reduction in the bandwidth means less data arrives at a given time, and so the total data sent is also reduced (with respect to time as iperf2 gives a 10 second time interval). Lastly, as the delay increases the initial round trip time is increased. This is an obvious effect of increasing the delay since irtt is an estimation of how long it takes for data to go from the sender to the receiver and back, so adding delay increases the the irtt since the time taken increases. From the measurements made a pattern can be seen. A delay of 20ms causes an irtt of around 10000 and a delay of 20ms causes an irtt of around 20000.

Furthermore, there are some interesting trends in the behaviour of TCP which can be observed. Looking at the amount by which the bandwidth is decreased between successive delay levels shows something interesting. Once the added delay reaches a certain quantity, 20ms based on the measurements, further increases in delay reduces the amount by which the bandwidth drops. This almost suggests an asymptotic behaviour. Another interesting observation is that all the measurements made consistently show that once the delay reaches around 20ms, the amount of data sent is below 1.1 gigabytes which is the default amount that is sent when using iperf2. So, not all the data that was meant to be sent is sent. All these measurements suggest that after a delay of 20ms, TCP’s performance in general significantly degrades which is indicated by the large drops in both the data transferred and bandwidth.

Conclusion

In conclusion, the results show that an increase in delay causes a decrease in the bandwidth, a decrease in the data transferred and an increase in the initial round trip time. This suggests that the overall performance of TCP is affected by delay, where a large delay significantly degrades its performance.

Module code: CS4105
Matriculation number: 200017941

Data file list

The data/ sub-directory contains five further sub-directories:

- Scripts:
  - no_delay.sh: bash script for running the experiment where there is no delay.
  - some_delay.sh: bash script for running the experiment where delay is a constant.
  - linear_increase_delay.sh: bash script for running the experiment where delay is increased linearly.
  - quadratic_increase_delay.sh: bash script for running the experiment where the delay is increased quadratically.
  - exponential_increase_delay.sh: bash script for running the experimetn where the delay is increased exponentially.
  - basic_stats.py: python code for getting the measurements and descriptive statistics tables as well as the graphs for measurements from no_delay.sh and some_delay.sh
  - basic_stats.py2: python code for getting the combined graph for all the measurements from no_delay.sh and some_delay.sh.
  - basic_stats.py3: python code for getting the measurements tables and the graphs for measurements from linear_increase_delay.sh, quadratic_increase_delay.sh and exponential_increase_delay.sh.
  - requirements.txt: contains the imports for the python files.
- CSVs: contains the measurements for each experiment in a CSV format which is the result of running iperf2.
- Tables: contains all the measurements and descriptive statistics tables from each experiment which.
- Graphs: contains all the graphs plotted for each experiment from the measurements stored in the CSVs.
- Screenshots: contains screenshots of the client’s and server’s terminal window after each type of experiment.