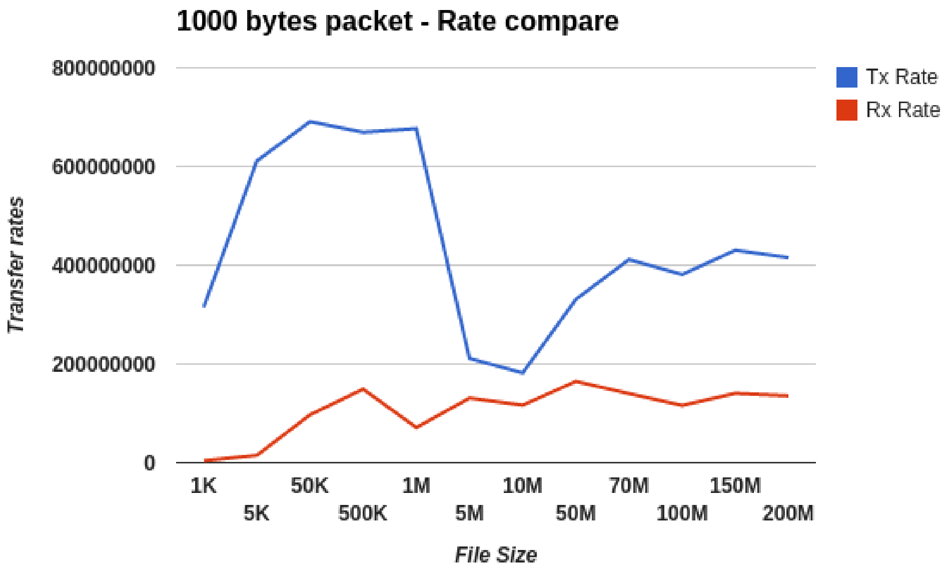
MNC Programming Assignment 1
Analysis Report

4.1 Data Rates vs File Size

(Constant packet size of 1000) the following observations were made before analysis.

File Size	Tx Rate	Rx Rate
1K	315076928	3876952
5K	611343232	14867513
50K	690725120	97038620
500K	669281088	148940032
1M	676745088	71164237
5M	211226560	130844223
10M	181753232	116535957
50M	330769280	164619400
70M	411472896	139909164
100M	381150144	116046783
150M	430454336	140347961
200M	415508544	135411407

Plotting the same data on a line chart:



- ❏ It can be observed that transmission rate(Tx) is always higher than the receiving rate(Rx) for all file sizes. This was not according to my expectations as I was expecting a similar transmission rate (or maybe a lower transmission rate because some links may have lower upload bandwidth).

  After analysis I this this can be attributed to various factors:
  - A chance that available download bandwidth is lesser than the upload bandwidth
  - The no. of computing steps between calculating receiving rate at Rx ends is higher and this reduces the rate at which *recv* is called and hence a lower rate
  - Another factor can be that socket's recv buffer size might be set at a lower value than the sending socket and hence the receive might have to be called more than once to receive the same packet. (This can be solved by using *setsockopt* to set the *recv* buffer size to be of the same size or whichever size is optimum.) **I think this might be the most probable reason for the reduced receiver rate.**

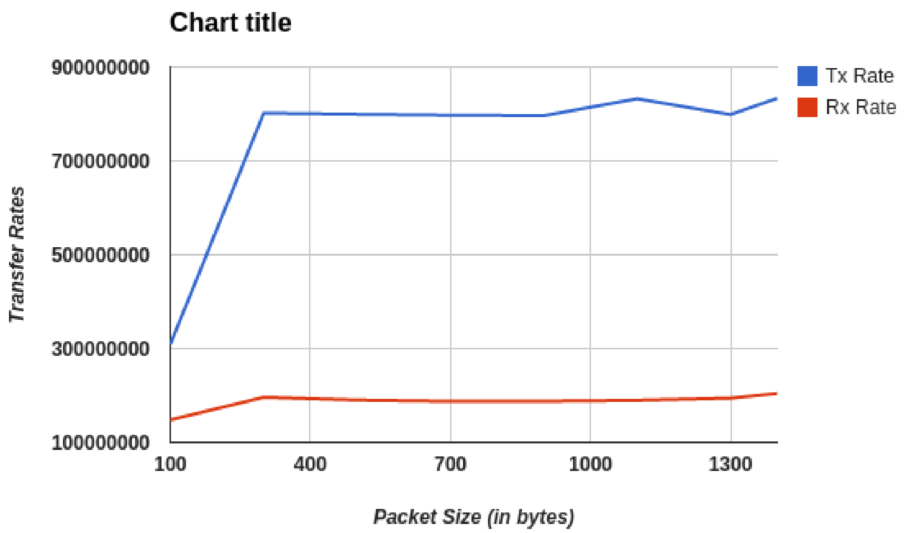
- ❏ Another observation is the significantly lower receiving rate at lower file sizes. This is as expected as at lower file sizes there might not be enough data to be transferred to make use of the entire bandwidth of the link.
- ❏ The lower transmission rates at 10 MB and 5 MB files seems to be as a result of the network load when the measurement was taken as later measurement showed regular transmission and reception rates.

4.2 Data Rates vs Packet Size

(Constant file size of 200 MB) the following observations were made before analysis:

Packet Size (bytes)	Tx Rate	Rx Rate
100	309338592	147870208
300	801898048	196043685
500	799547072	190172683
700	797499840	187553176
900	796463680	187782669
1100	832555392	189648818
1300	798908288	194209548
1400	833510464	204220657

Plotting the same data on a line chart:



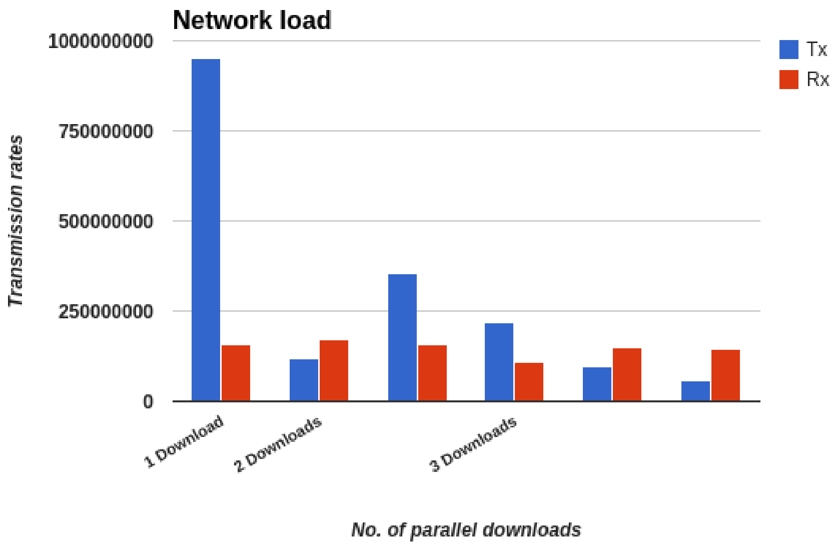
- It can be observed that the previous trend of low reception rates continues for different packet sizes also. The same arguments from above applies here also
- Another observation is the low transmission rate at small packet sizes. This was expected at arbitrary lower packet sizes. This can be attributed to the fact that that send will have to be called multiple times to send the same amount of data. (This is true for higher chunk sizes also when talking relatively) However increasing packet size beyond a point (when the bandwidth is utilized fully) doesn't affect the transmission rate since *send* function can only send the maximum no of bytes limited by the link bandwidth. Increasing packet size beyond the maximum capacity becomes a waste as unnecessarily memory will have to be allocated during the transmission time.

4.3 Data Rates vs Load variations

(For constant file size of 70 MB and packet size of 1000 bytes) the following observations were made before analysis:

Parallel Downloads	Tx	Rx
1 Download	952730560	158255950
2 Downloads	119288688	172210998
	354278432	156986160
3 Downloads	220818096	107889945
	97662120	148182968
	57836412	145945419

Plotting the same data on a bar chart:



- It can be observed that the transmission rates are divided (and reduced) when the number of parallel downloads increases. This is as expected because the bandwidth will be divided between the links and the transmitter will have to retransmit the dropped packets sent to the receiver (this happens when receiver buffer is full with data from the parallel connection)
- The reception rate is same throughout as although data is being sent parallely from the transmitter, according to the design of the application which ever socket has data first will be written to file and then the next and so on. Since the receiving rate is calculated based on time of start of receiving the first packet to time of file written to disk the receiving rate remains the same.
- It can be noted that sum of transmission rates in parallel connections doesn't match up to the transmission rate of the individual connection or (Tx1 > Sum($Tx2_a$ + $Tx2_b$) > Sum ($Tx3_a$+$Tx3_b$+$Tx3_c$)) . This can be accounted by the time required for retransmission of packets because of dropped packets as discussed earlier.

The analysis till now shows a better design of the application could have increase the transmission rates during file transfer.

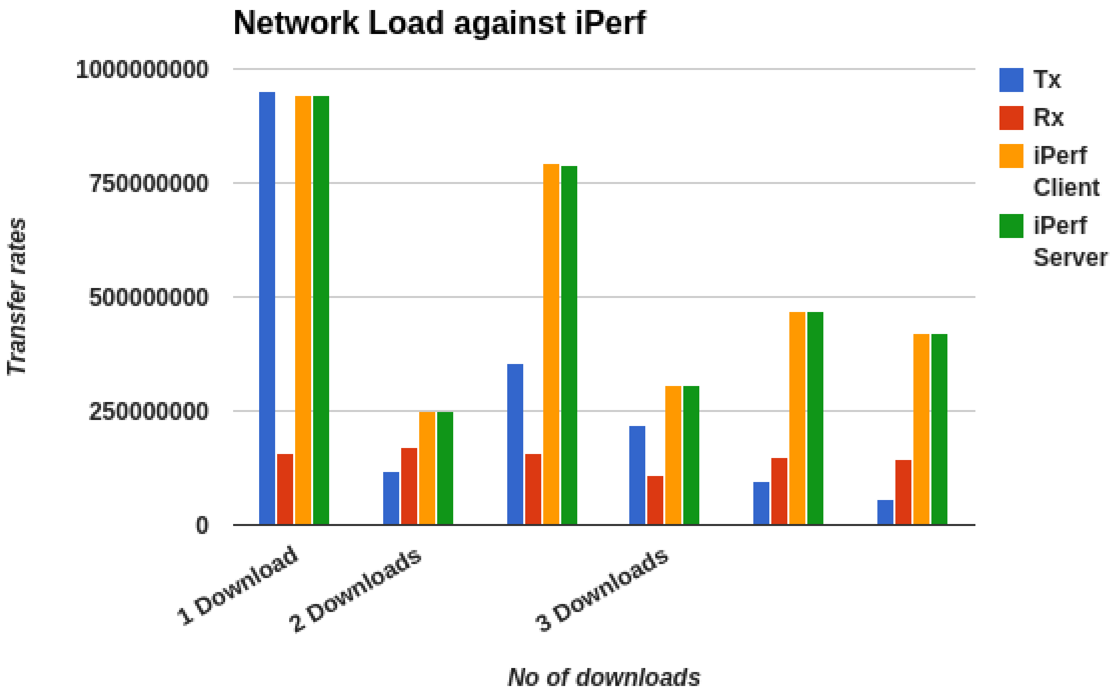
4.4 iPerf and Network Bandwidth

Measurements were made with iperf for different network loads and compared against section 4.3. It is as follows:

	Tx	Rx	iPerf Client	iPerf Server

1 Download	952730560	158255950	944000000	941000000
2 Downloads	119288688	172210998	251000000	251000000
	354278432	156986160	793000000	791000000
3 Downloads	220818096	107889945	307000000	305000000
	97662120	148182968	471000000	470000000
	57836412	145945419	420000000	419000000

The same data was plotted using a bar chart:



- ➔ It can be noted that except in the case of transmission rate of single download iPerf measurements indicates much higher transmission and reception rates in all other cases.
- ➔ After the analysis, I think, expected and actual performance was different mainly due to three reasons:
  - ◆ Not taking care of send and receiver buffer of socket and setting it according to the available bandwidth of the link
  - ◆ Not matching the receive buffer according to the send buffer
  - ◆ Even though multiplexed the *recv* for parallel downloads behaves serially due to flaw in design of the application