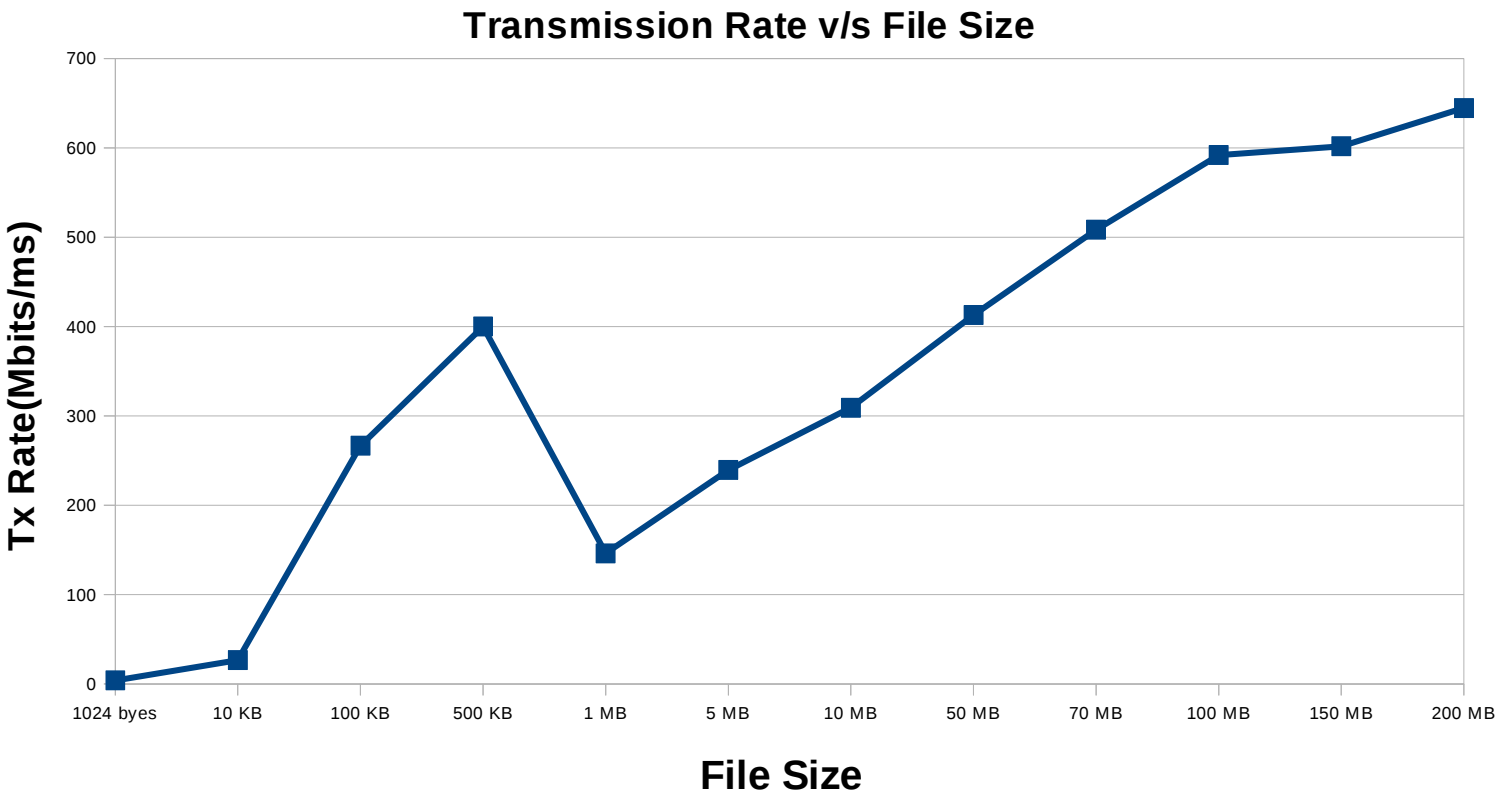
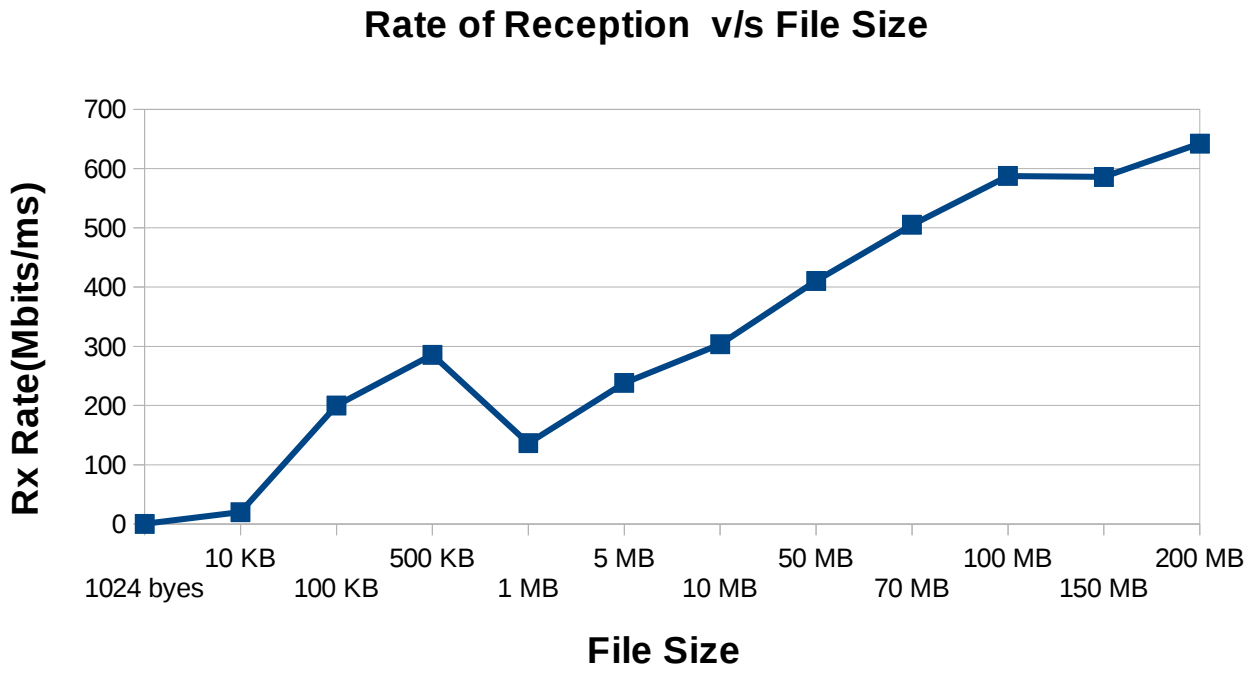
1) Data Rates vs. File Size

As we can see from the graph the Rate of reception (download speed) for the file of size 1KB comes out to be more as compared to the download speed of rest of the files. The possible explanation of this behavior is that the file handling operations and insufficient data for the file transmission is taking up more time.

The transmission rate increases till the file size of 500KB. After that there was dip at file of size 1 MB. The reason for this inconsistency could be the network congestion at the instant when the file transfer was carried. I am also suspecting this to be the reason as the Rx/Tx rates increases consistently with the file size (till 200 MB).

The reason for the consistent increase in the Tx/Rx rates with file size is we are more efficiently utilizing the available bandwidth. Also the time to perform the file handling operations is undermined here by the efficient utilization of the bandwidth. Also the packet size of 1000 Byes seems to ideal for the file transfer operations (as can be seen in part 2).

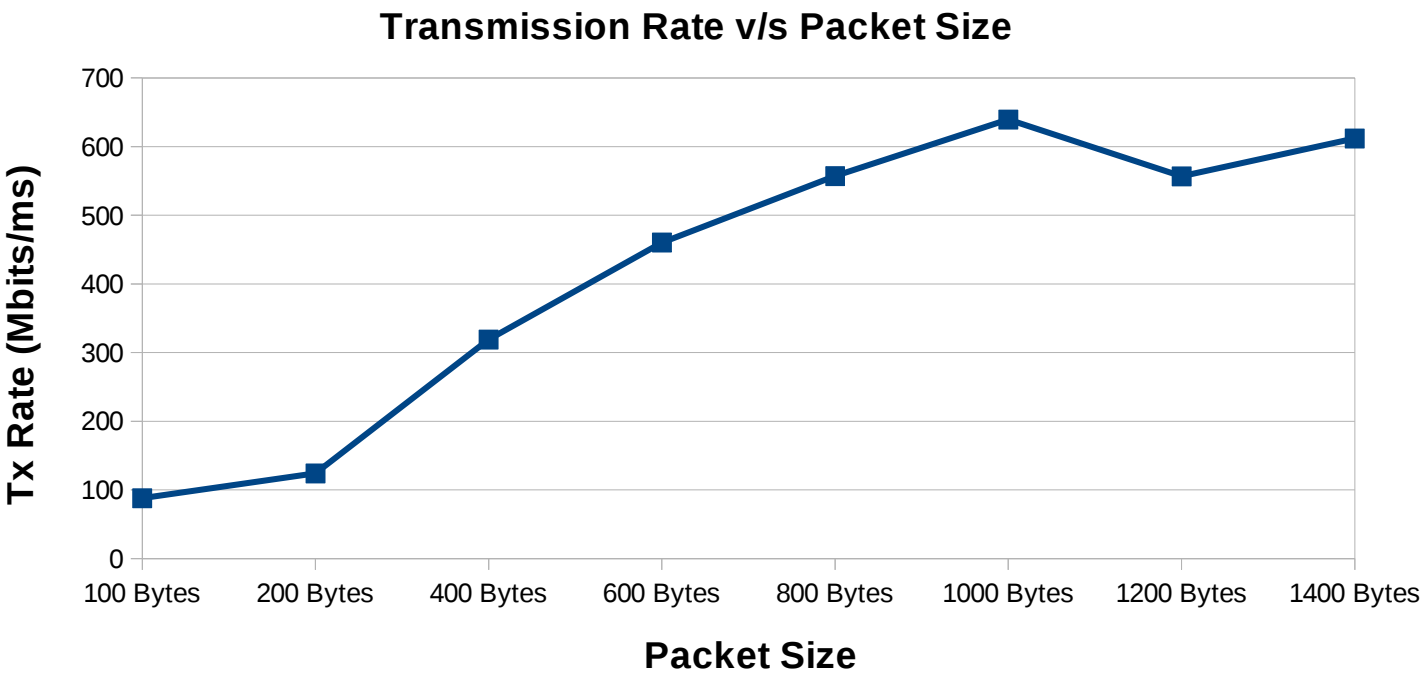


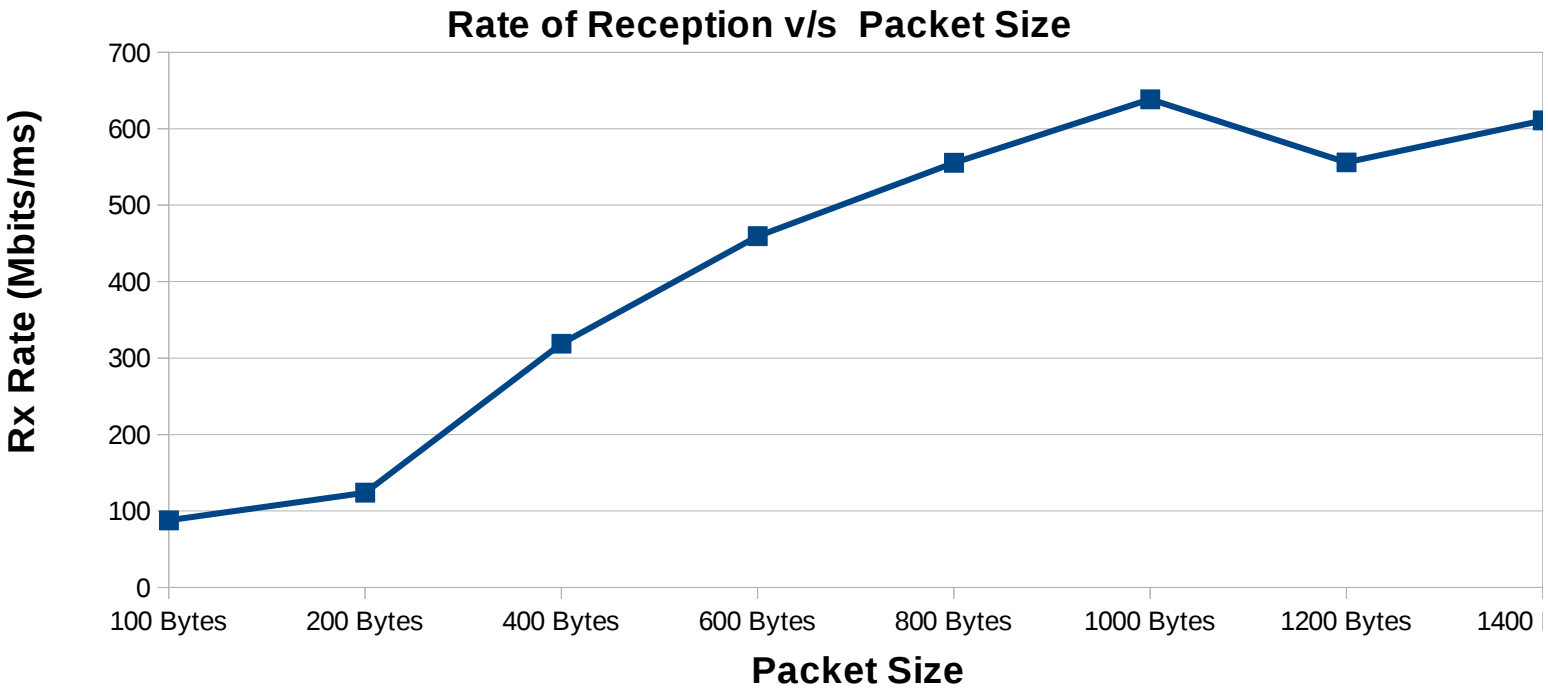


2) Data Rates vs. Packet Size

In this case we are varying the packet size while keeping the file size constant. As we can see from the graphs for this case (shown below) for Tx and Rx, the file transfer rates increased consistently as we increase the packet size till 1000 Bytes. The reason for the increase in the file transfer rate is that we are sending/receiving more data per send() / recv() call. This results in the efficient utilization of the available bandwidth and reduces the total overhead incurred in the send()/recv() operation. Also at the receiving end we will be writing more data to the file per recv. Similarly at the sending end, we are reading more data from the file per send. This results in the reduction of the number of reads and writes required for the file transfer operations. Hence we are saving the time here. So overall the performance of the file tranfer increases with the packet size till 1000 Bytes

We can observe the dip in the Tx/Rx Rates when the packet size is more that 1000 Bytes. The possible reason for this dip is the oversized packets which are resulting in excess buffering of packets which results in TCP slowing down the transmission rate. In this case this latency incurred due to the over sized packets outweighs the advantage gained due to the reduction in file read/write operations.


Transmission Rate v/s Packet Size
Tx Rate (Mbits/ms)
700
600
500
400
300
200
100
0
100 Bytes
200 Bytes
400 Bytes
600 Bytes
800 Bytes
1000 Bytes
1200 Bytes
1400 Bytes
Packet Size


Rate of Reception v/s Packet Size
Rx Rate (Mbits/ms)
700
600
500
400
300
200
100
0
100 Bytes
200 Bytes
400 Bytes
600 Bytes
800 Bytes
1000 Bytes
1200 Bytes
1400
Packet Size

4.3 Data Rates vs. Load Variations

The expected behavior in case of parallel downloads/upload operation on any machine is the division of bandwidth between the operation. This is what we can see when we run the iperf and vary the number of clients performing the file transmission to the server.
When we run the iperf as server (i.e. with -s option) and there is only one client connecting to it than we can see the bandwidth utilization comes at about 943 Mbps. This is approximately equal to the full available bandwidth of 1Gbps.
In the case of two clients the bandwidth gets divided between the two as the server can serve up to 1 Gbps. In this case we can see one of the client get the speed as 785 Mbps and the other gets the speed of 261 Mbps which is approximately equal to the servers capacity.
When we have three clients connecting to the server using iperf we can see the approximate divison of server's bandwidth between the clients. This can be verified from the screeshot in the iperf section.

The similar effects can be seen in our case also. In our scenario we are downloading different files from multiple peers. In case when there is one file downloaded from a single peer the data is

Tx Time (in ms): 935	Rx Time (in ms): 940
Tx Rate (Mbps) : 613 .3	Rx Rate (Mbps) : 610.04

In case there are downloads from 2 clients:

Statistics for the receiving end:

Rx:euston.cse.buffalo.edu <- highgate.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 1039 millisecond, Rx Rate: 565161 bits/millisecond

Rx:euston.cse.buffalo.edu <- embankment.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 8793 millisecond, Rx Rate: 66780 bits/millisecond

Stats for the transmitting end:

1) Tx:highgate.cse.buffalo.edu -> euston.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 1034 milliseconds, Tx Rate: 567894 bits/milliseconds

2) Tx:embankment.cse.buffalo.edu -> euston.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 8791 milliseconds, Tx Rate: 66795 bits/milliseconds

Analysis:
In this case we can see the behavior similar to that observed in the case of iperf. The downloading clients bandwidth is divided between the two connections.
We also observer the change in the mismatch in the upload rate and the dowload rate. The main reason for this is that the send() function is non blocking. It justs goes through all its sends. But the downloading entity is not able to recv() all the packets and the packets sent are waiting. Again the divison of bandwidth due to parallel download is the reason for this delay.

In the case of downloads from 3 clients:

Statistics of the receving end.
Rx:euston.cse.buffalo.edu <- highgate.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 2352 millisecond, Rx Rate: 249660 bits/millisecond

Rx:euston.cse.buffalo.edu <- embankment.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 12914 millisecond, Rx Rate: 45470 bits/millisecond

Rx:euston.cse.buffalo.edu <- underground.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 12866 millisecond, Rx Rate: 45639 bits/millisecond

Statistics of the transmitting end:

1) Tx:highgate.cse.buffalo.edu -> euston.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 2336 milliseconds, Tx Rate: 251370 bits/milliseconds

2) Tx:embankment.cse.buffalo.edu -> euston.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 2348 milliseconds, Tx Rate: 250086 bits/milliseconds

3) Tx:underground.cse.buffalo.edu -> euston.cse.buffalo.edu, File Size: 73400320 Bytes, Time Taken: 2308 milliseconds, Tx Rate: 254420 bits/milliseconds.

Analysis:
In this case we can see the behavior similar to that observed in the 2nd case of iperf. The receiving ends bandwidth is divided between the 3 connections.

4) Network Bandwidth Measurement using iperf:

The analysis and comparison for this section has been done in the section 3) above.

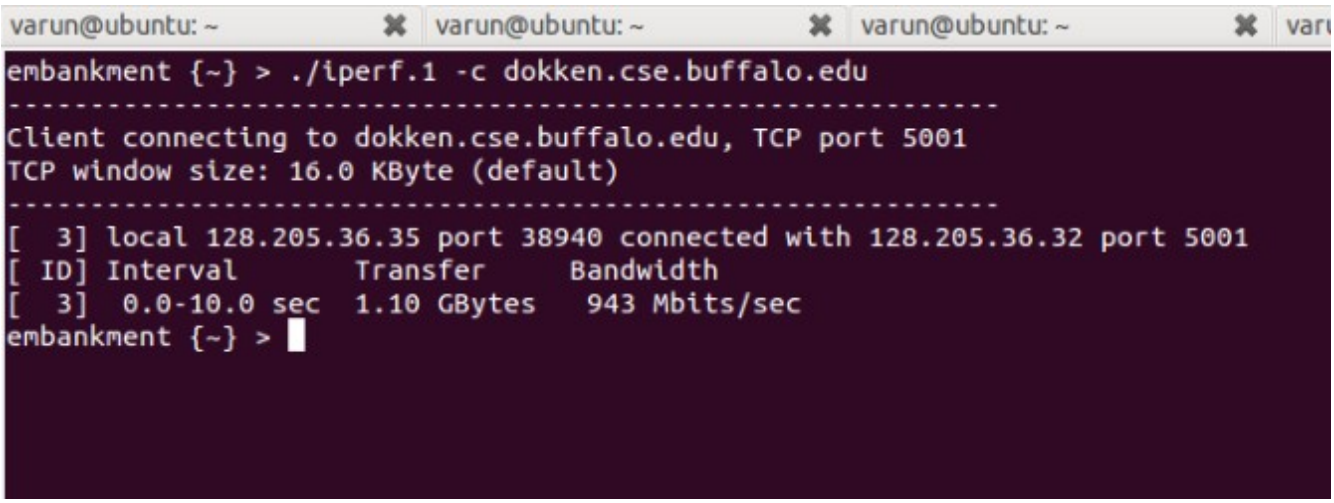
The observed network b/w for the single client case is 943 Mbps which is approximately equal to the available b/w of 1Gbps. This cane be seen from the screenshot **iperf: Single Client Case**

The observed b/w for the two clients case:
Client 1: 261 Mbps
Client 2: 785 Mbps.

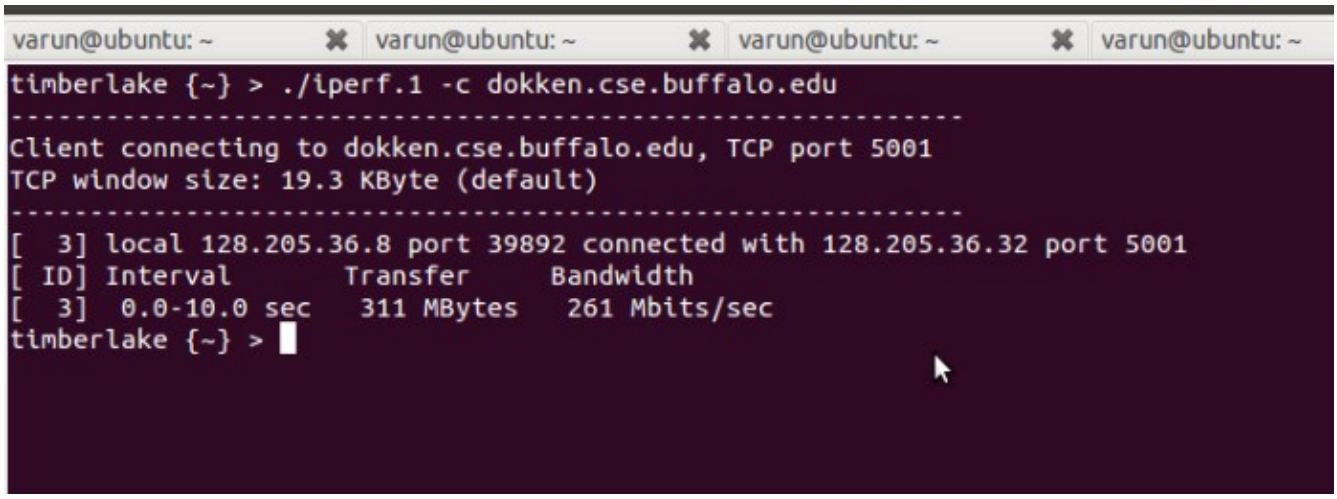
The possible reason the total b/w exceeding 1 Gbps could be that the 2nd client was able to get more B/W when the first client was done. Even then the total b/w is approximately equal to 1Gbps.
Please refer the screenshot **iperf : Two Clients Case – 1, iperf : Two Clients Case – 2**

The observed b/w for the three clients case:
Client 1: 393 Mbps
Client 2: 392 Mbps.
Client 3: 341 Mbps

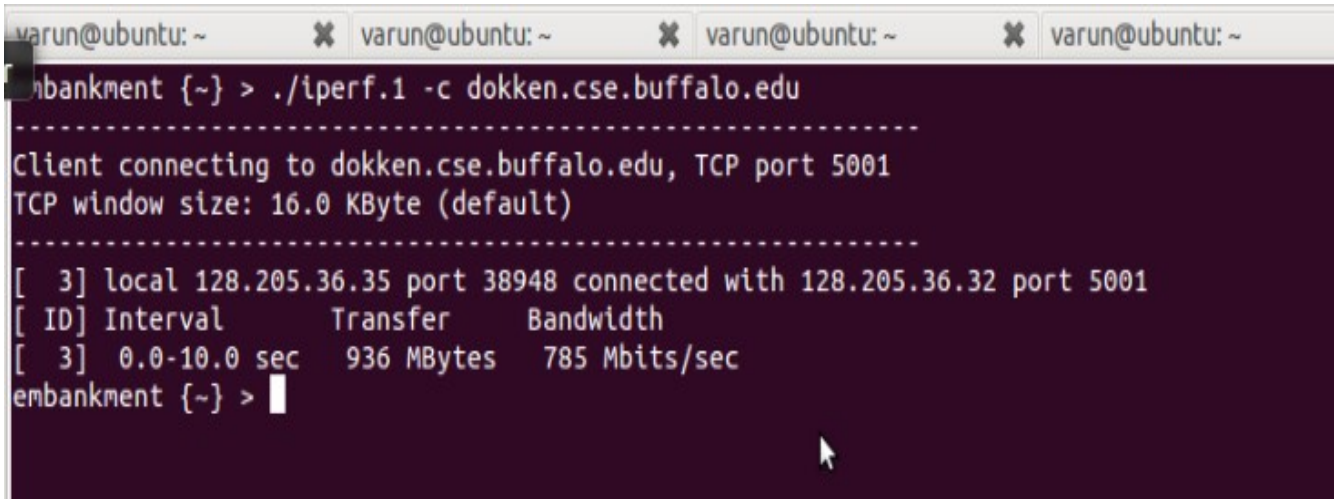
Total Bandwidth in this case is slightly more that 1 Gbps. The reason for this is same as for the two client case.



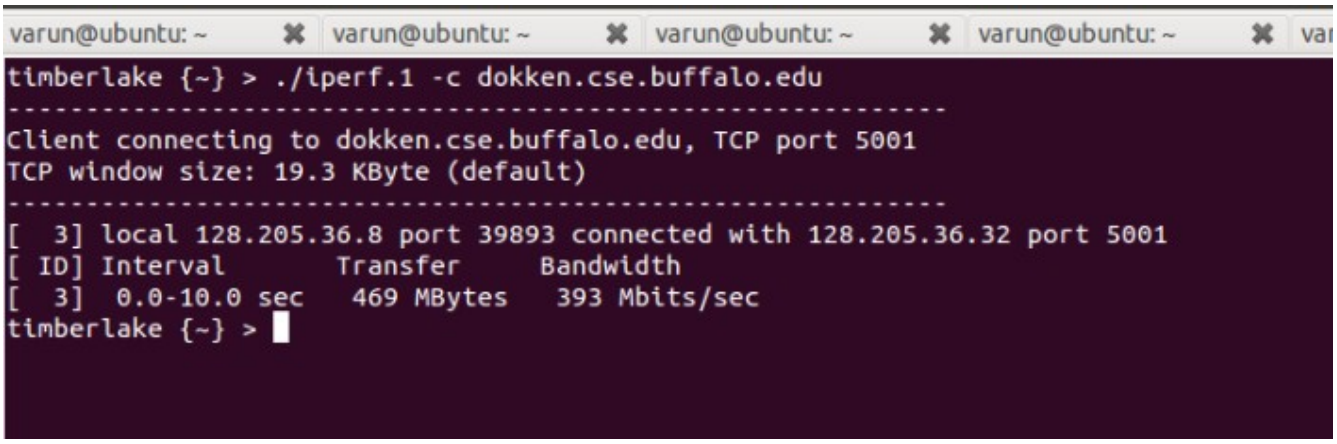
iperf: Single Client Case



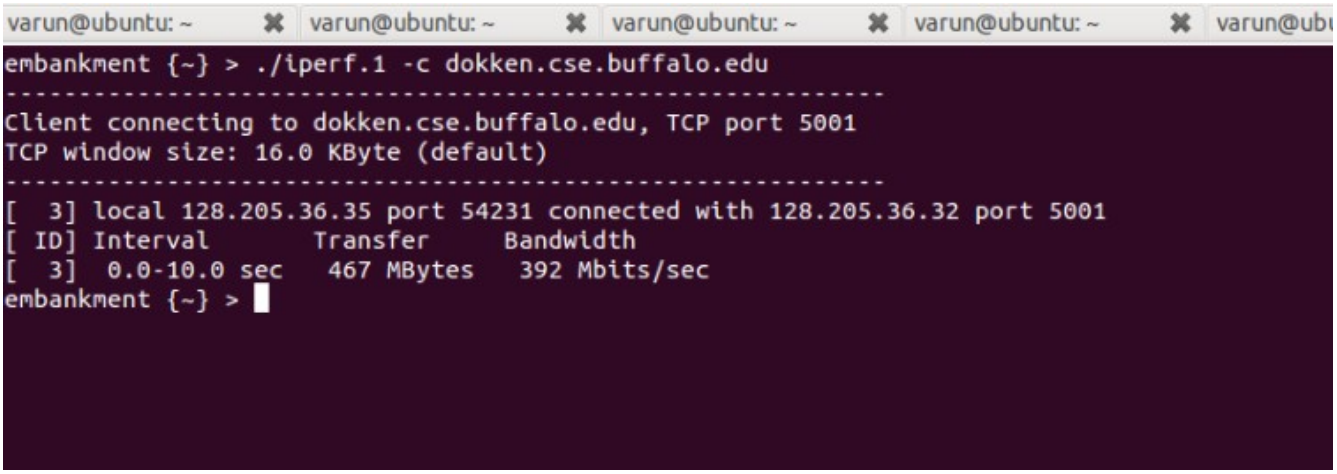
iperf : Two Clients Case – 1



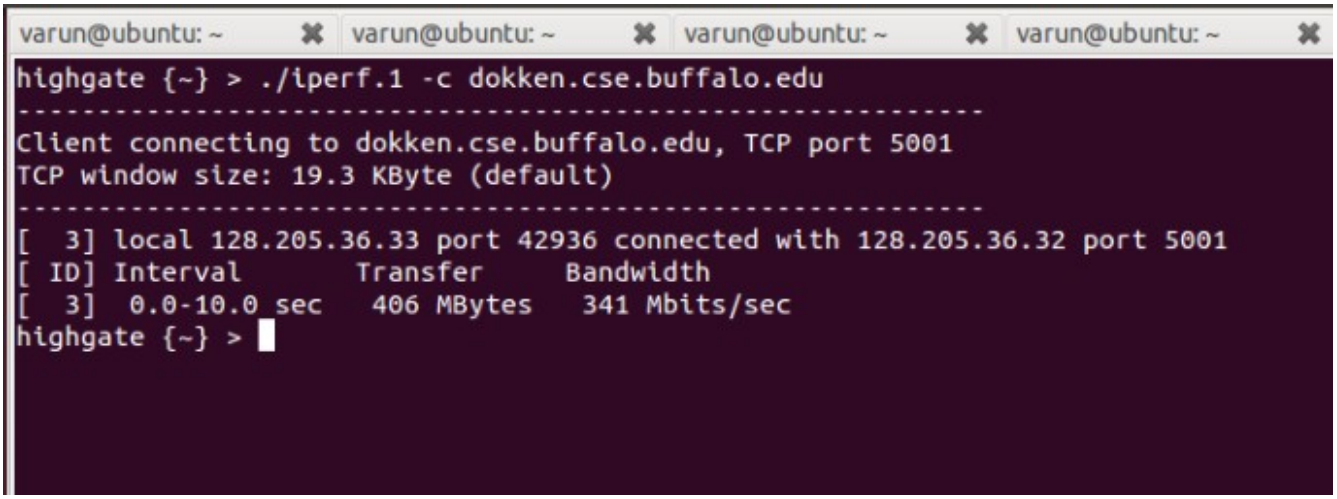
iperf – Two Clients Case – 2



iperf – Three Clients Case – 1



iperf – Three Clients Case – 2



iperf – Three Clients Case – 2