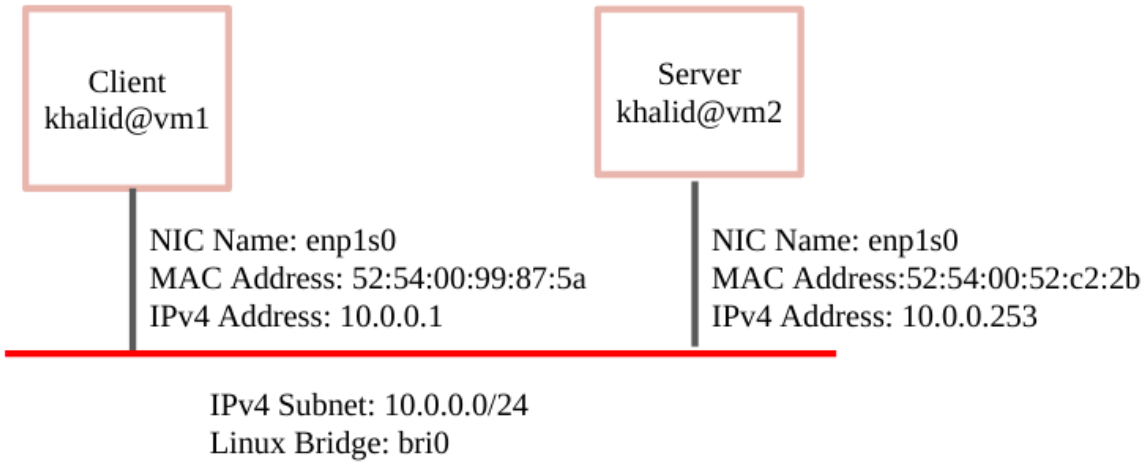
CS3543 Lab Assignment 2

Member 1: Akash Tadwai (ES18BTECH11019)
Member 2: Khalid Shareef (CS18BTECH11029)
Member 3: Sai Anuraag Marapally (CS18BTECH11027)
Member 4: Vinta Reethu (ES18BTECH11028)

Network Diagram



Task #1

Part #1

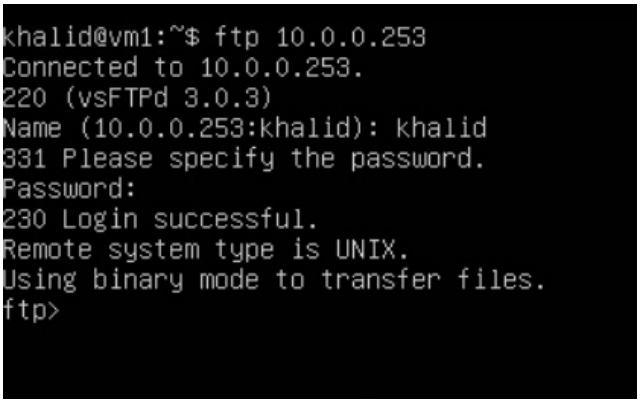
In order to Configure 100Mbit link speed on both VMs we have used the "**tc**" command. We have set the two NIC interfaces (both **enp1s0**) to the traffic rate of 100Mbit. We set the rate for both client and server and show the parameters in the below screenshots.

```
khalid@vm1:~$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
[sudo] password for khalid:
khalid@vm1:~$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
khalid@vm1:~$ _
```

```
khalid@vm2:~$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
[sudo] password for khalid:
khalid@vm2:~$ tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
khalid@vm2:~$ _
```

Next for FTP transfer we have installed vsFTPd and logged into FTP in VM1 and transferred the 100MB files from VM1 to VM2.

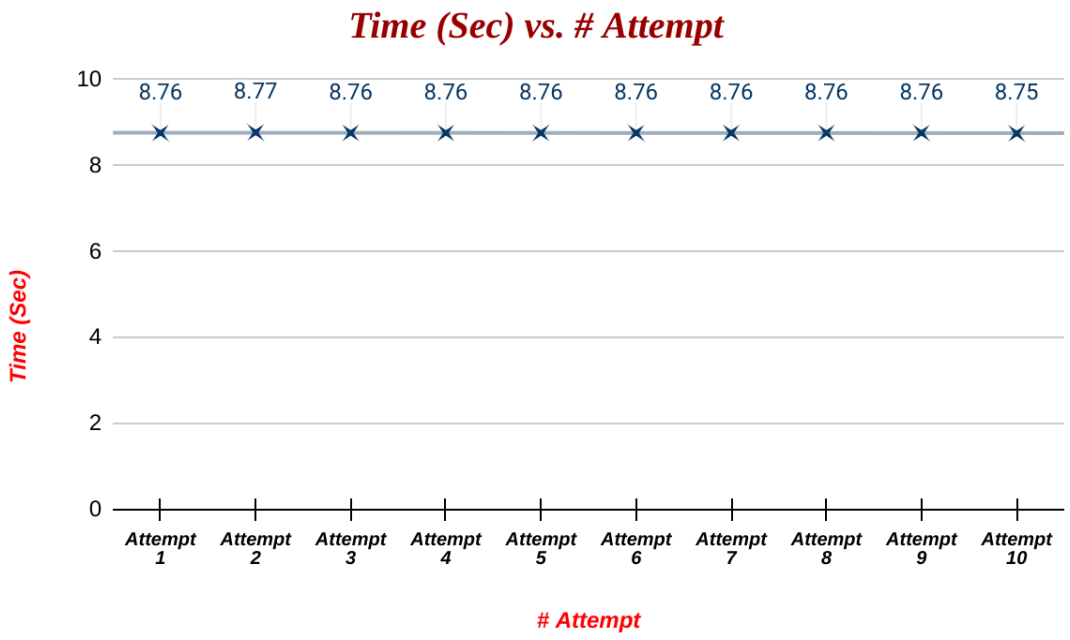
FTP Login in VM1:

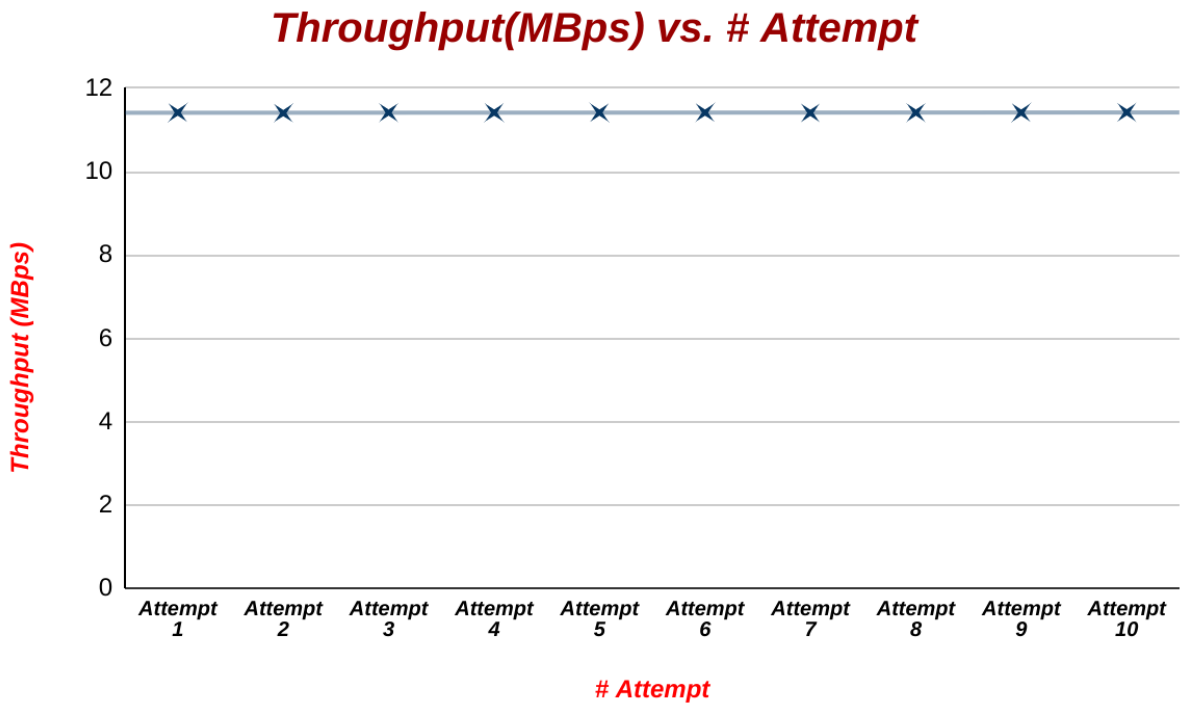


FTP File Transfer:

Table and Graphs representing transfer of 100MB file without loss and delay over 100Mbit

Without delay and packet loss	Time (Sec)	Throughput (MBps)
Attempt 1	8.76	11.4147
Attempt 2	8.77	11.4064
Attempt 3	8.76	11.418
Attempt 4	8.76	11.4187
Attempt 5	8.76	11.4146
Attempt 6	8.76	11.4188
Attempt 7	8.76	11.4113
Attempt 8	8.76	11.4173
Attempt 9	8.76	11.4142
Attempt 10	8.75	11.4246
Average	**8.76**	**11.41586**





Justification :

As we can observe that, the average effective bandwidth is 11.41586 MB/s which is approximately **91.326 Mbps** which is close to the link rate 100 Mbps that we have set. This is because we have not added ***any loss or delay*** explicitly using tc command.
Also, we can calculate the average time as
average time taken = ***File size / average effective bandwidth***
= 100 MB / (11.41586MB/s)
= **8.7597 secs**
This is consistent with the observed average time taken **8.76 secs**.

Part #2

In this part we modify the existing rule and add a delay of 50 ms and loss of 5% to both the interfaces in client and server and check the qdisc configuration.

Client:

```
khalid@vm1:~$ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms loss 5%
khalid@vm1:~$ tc qdisc show dev enp1s0
qdisc netem 8001: root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
khalid@vm1:~$
```

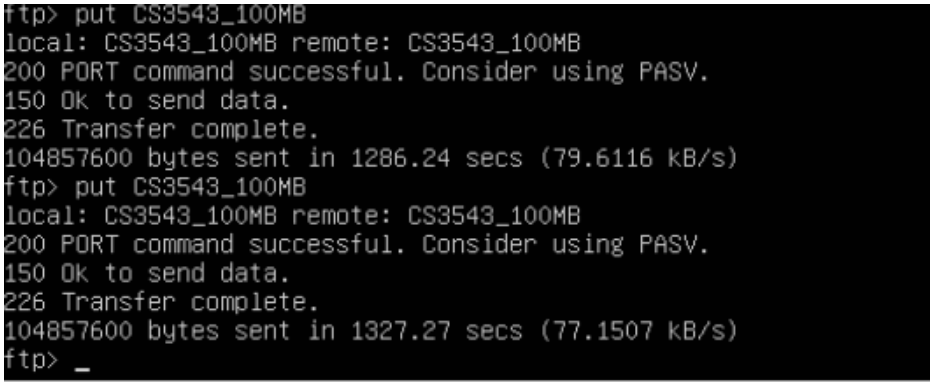
Server:

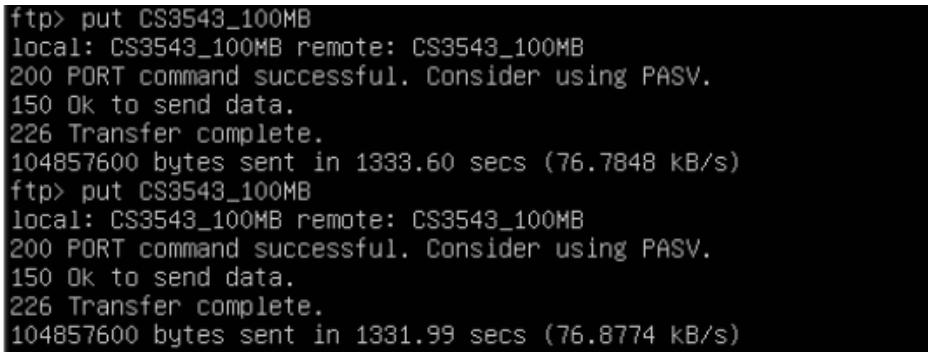
```
khalid@vm2:~$ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms loss 5%
khalid@vm2:~$ tc qdisc show dev enp1s0
qdisc netem 8001: root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
khalid@vm2:~$ _
```

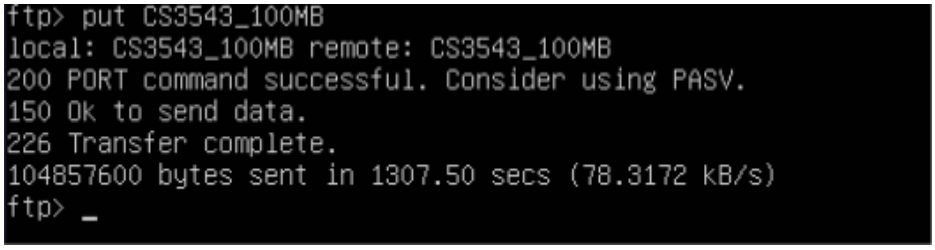
FTP File Transfer:

```
ftp> put CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
104857600 bytes sent in 1283.26 secs (79.7965 kB/s)
ftp> put CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
104857600 bytes sent in 1299.54 secs (78.7974 kB/s)
ftp> _
```

```
ftp> put CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
104857600 bytes sent in 1316.80 secs (77.7643 kB/s)
ftp> put CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
104857600 bytes sent in 1330.25 secs (76.9782 kB/s)
```







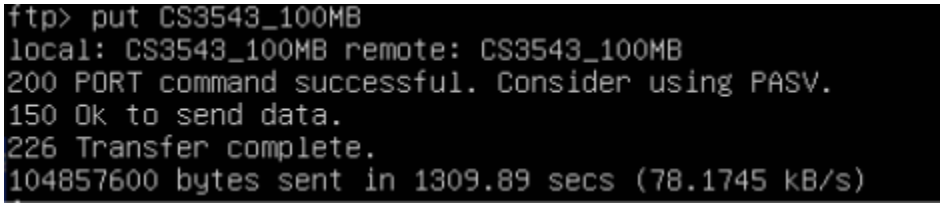


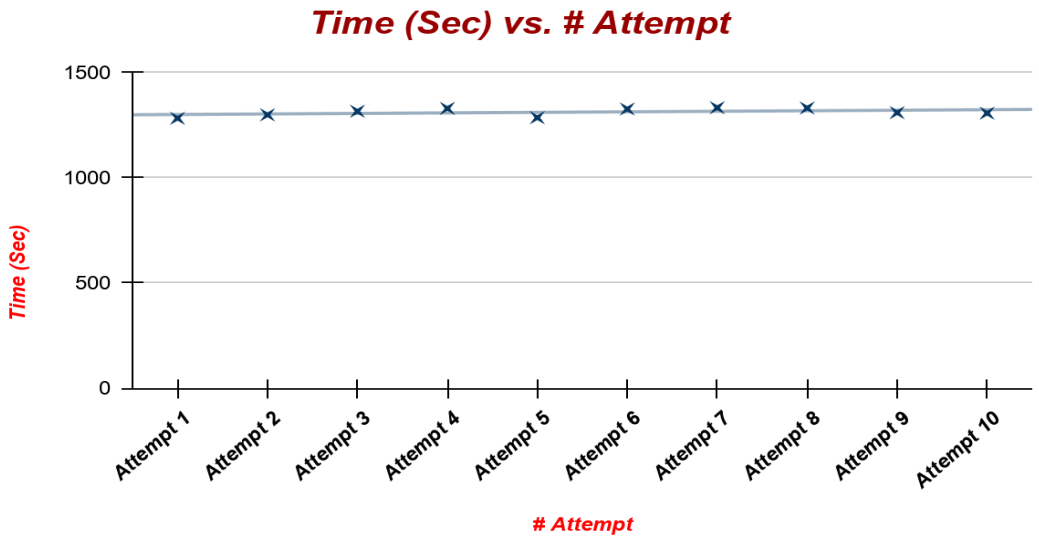
Table and Graphs representing transfer of 100MB file with 5% loss and 50ms delay over 100Mbit Link

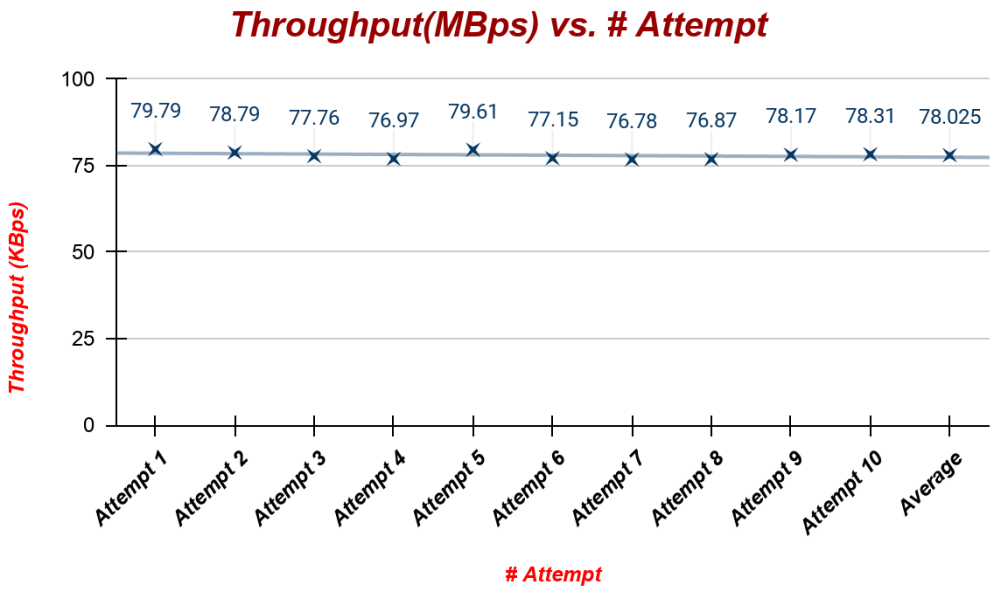
With delay, packet loss	Time (Sec)	Throughput (KBps)
Attempt 1	1283.26	79.7965
Attempt 2	1299.54	78.7984
Attempt 3	1316.8	77.7643
Attempt 4	1330.25	76.9782

Attempt 5	1286.24	79.6116
Attempt 6	1327.27	77.1507
Attempt 7	1333.6	76.7848
Attempt 8	1331.99	76.8774
Attempt 9	1309.89	78.1745
Attempt 10	1307.5	78.3172
Average	**1312.634**	**78.02536**

Justification :

The average effective throughput is **70.025 KBps**. This is because on each interface 100ms delay and 5% packet loss is added in its RTT. Due to this delay and packet loss there might be more retransmissions and hence throughput is highly reduced.





Task #2

In this Task we have implemented a simple FTP using UDP. Our Packet structure is as follows,

Packet Format

```
0                   1                   2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          SeqNumber            |          DataSize             |          Payload              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SeqNumber - 4 bytes (32bits)
DataSize - 4 bytes (32bits)
PayLoad - 4088 bytes

Total Packet Size - 4096 bytes

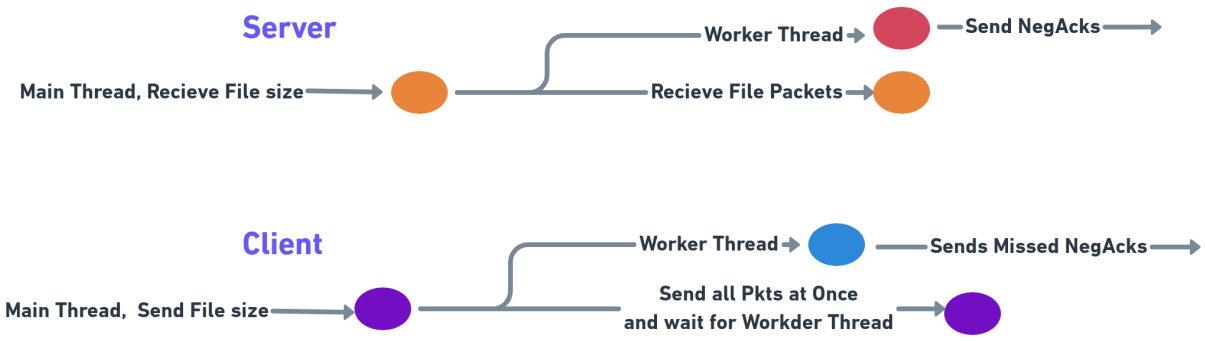
Special Features

- PacketLoss detection
- NAcks for packets which are not received yet
- High Throughput in lossy environment

Algorithm & Implementation

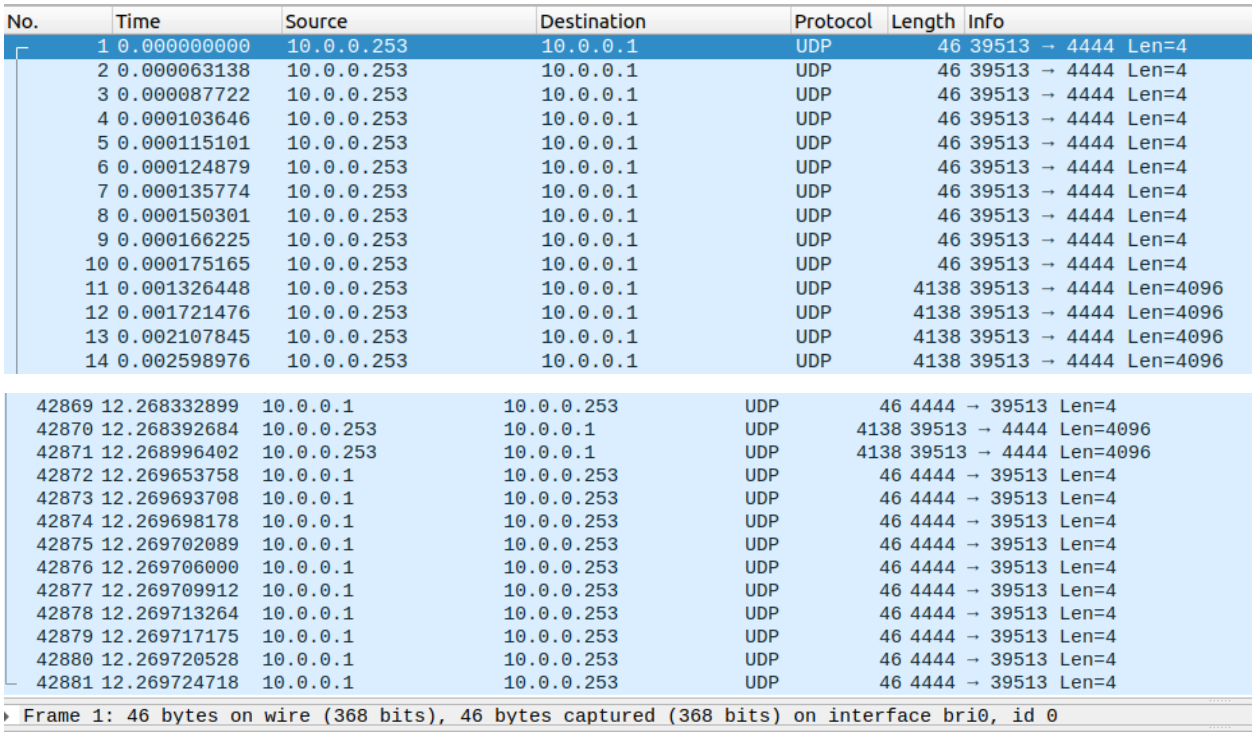
1. Initiate UDP sockets on both Client and Server sides.
2. Server and Client both create a Child Thread each.
3. Client sends the size of file first.
4. Server will calculate the number of packets from the file size and initializes a packet status array, which marks whether a packet is received or not.
5. While the main thread of the client sends each packet to the server, the main thread of the server marks whether the packet is received or not in the packet status array.
6. The child thread of the server iterates through the packet status array and if it finds whether any packet is not yet received it sends a request to the client asking for the packet.
7. The child thread of the client responds to the missing packet requests and sends those packets back to the server.

Pictorial Representation of Algorithm



Part #1

Below are the screen captures from wireshark before adding loss with link speed set to 100Mbit. We can see that the time is starting from **0.000 sec** and the file transfer ended at **12.26 sec.**



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
2	0.000063138	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
3	0.000087722	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
4	0.000103646	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
5	0.000115101	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
6	0.000124879	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
7	0.000135774	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
8	0.000150301	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
9	0.000166225	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
10	0.000175165	10.0.0.253	10.0.0.1	UDP	46	39513 → 4444 Len=4
11	0.001326448	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
12	0.001721476	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
13	0.002107845	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
14	0.002598976	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
42869	12.268332899	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42870	12.268392684	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
42871	12.268996402	10.0.0.253	10.0.0.1	UDP	4138	39513 → 4444 Len=4096
42872	12.269653758	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42873	12.269693708	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42874	12.269698178	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42875	12.269702089	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42876	12.269706000	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42877	12.269709912	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42878	12.269713264	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42879	12.269717175	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42880	12.269720528	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4
42881	12.269724718	10.0.0.1	10.0.0.253	UDP	46	4444 → 39513 Len=4

Throughput calculated fromWireshark:

Throughput = ***File size / average effective bandwidth***

= 104857600 bytes / (12.26-0.0)sec

= **8.55282219 MBps**

Screen Capture of file transfer b/w VMs

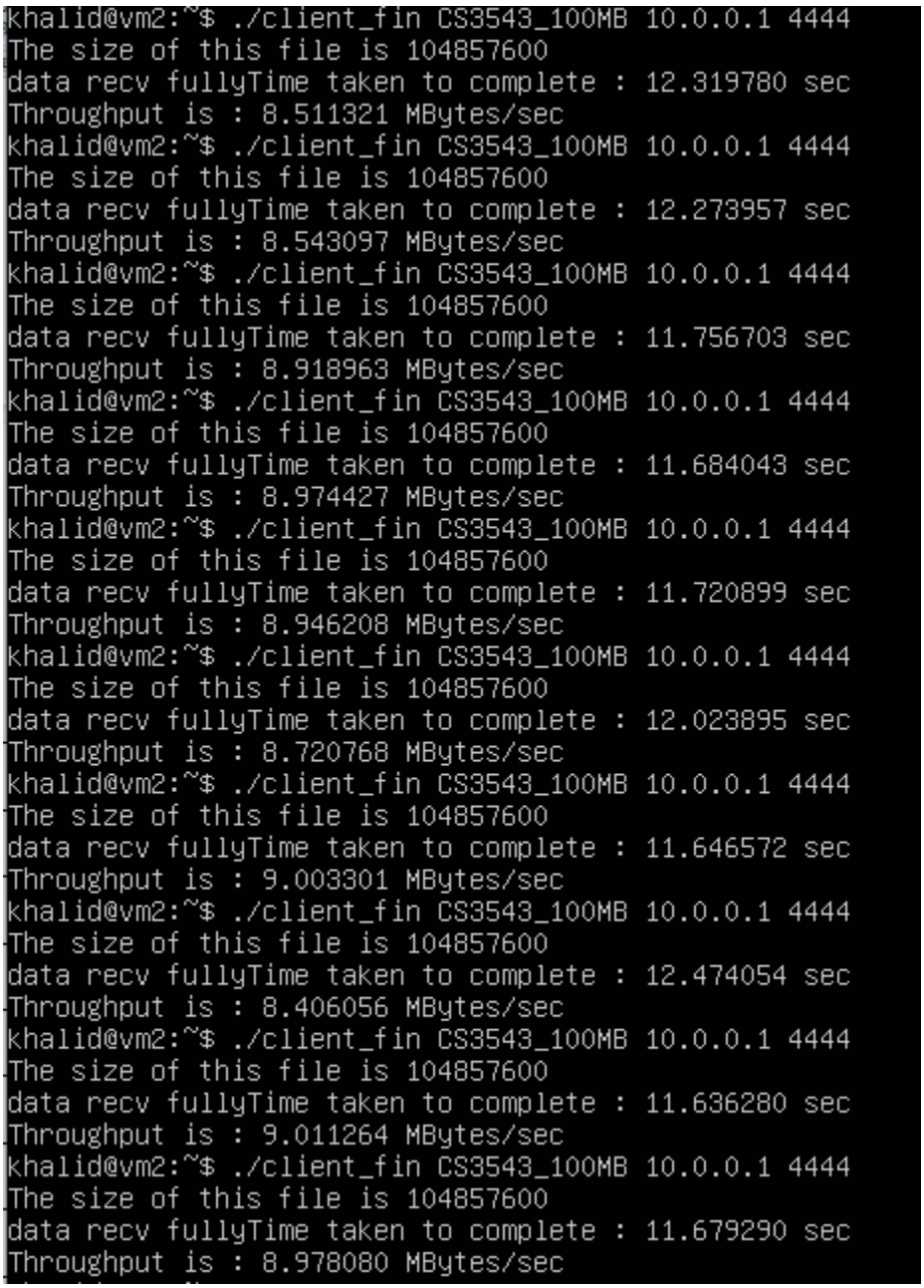
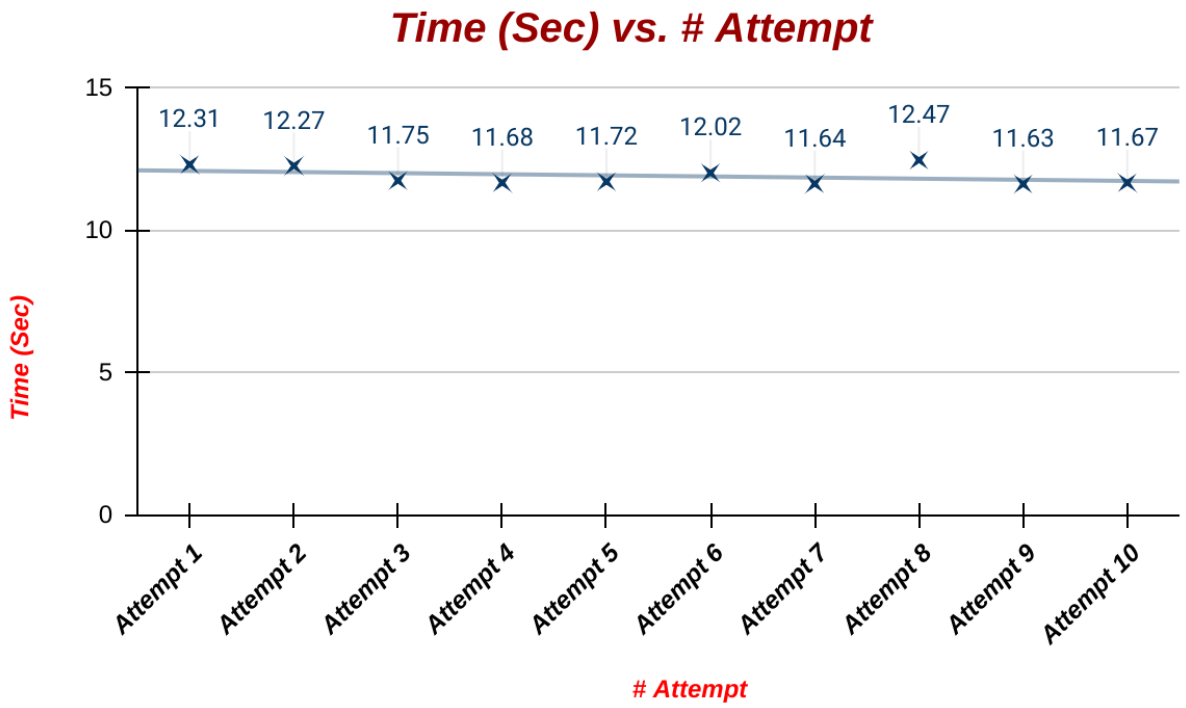
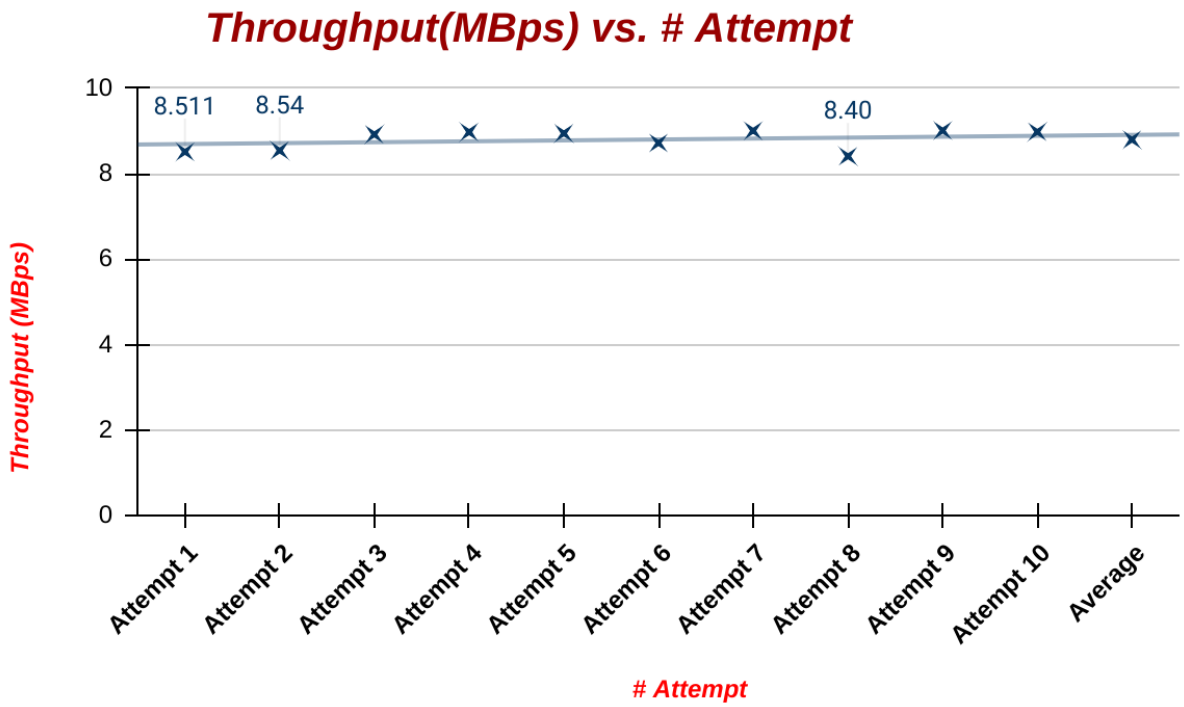


Table and Graphs representing transfer of 100MB file without loss and delay over 100Mbit

Without delay and packet loss	Time (Sec)	Throughput (MBps)
1	12.319780	8.511320819
2	12.273957	8.543096574
3	11.756703	8.91896308
4	11.684043	8.974427773
5	11.720899	8.946207966
6	12.023895	8.720768104
7	11.646572	9.003301572
8	12.474054	8.406056283
9	11.636280	9.011264768
10	11.679290	8.978080003
Average:	**11.9215473**	**8.795636788**

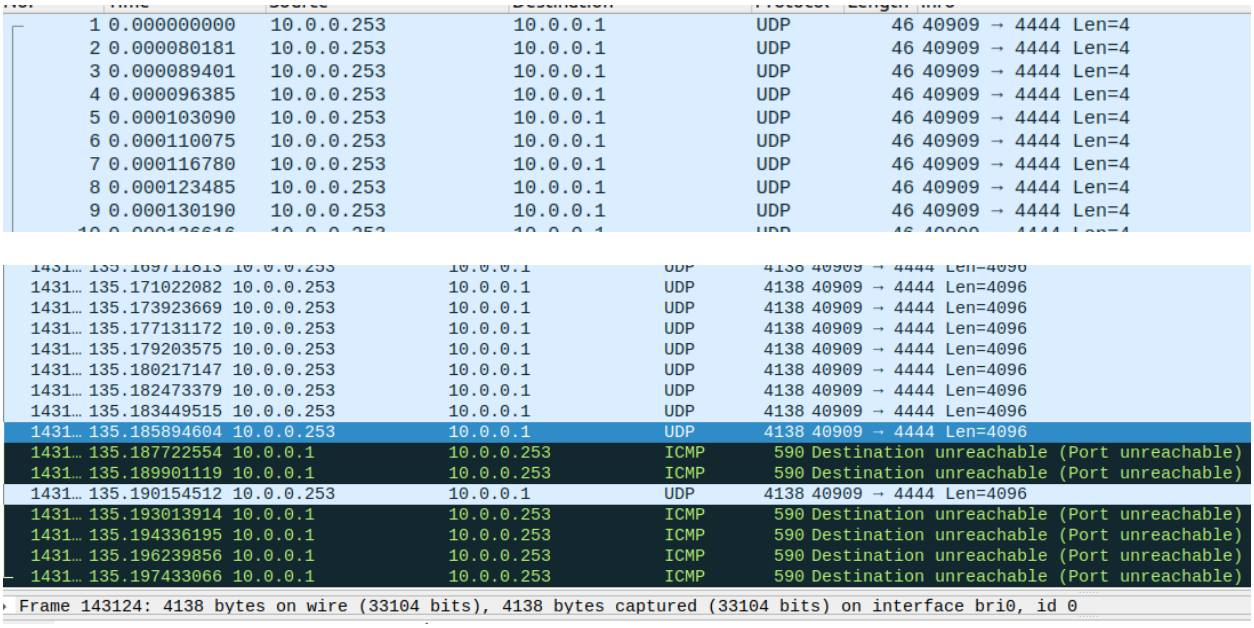




We can see that the average throughput of the File transfer for 10 runs is close to that of measured through wireshark packet capture. **(8.552 ~ 8.79)**

Part # 2

Below are the screen captures from wireshark after adding 50ms delay and 5% loss. We can see that the time is starting from 0.000 sec and the file transfer ended at 135.185 sec.



1	0.000000000	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
2	0.000080181	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
3	0.000089401	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
4	0.000096385	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
5	0.000103090	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
6	0.000110075	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
7	0.000116780	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
8	0.000123485	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4
9	0.000130190	10.0.0.253	10.0.0.1	UDP	46 40909 → 4444 Len=4

1431…	135.171022082	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.173923669	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.177131172	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.179203575	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.180217147	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.182473379	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.183449515	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.185894604	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.187722554	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)
1431…	135.189901119	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)
1431…	135.190154512	10.0.0.253	10.0.0.1	UDP	4138 40909 → 4444 Len=4096
1431…	135.193013914	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)
1431…	135.194336195	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)
1431…	135.196239856	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)
1431…	135.197433066	10.0.0.1	10.0.0.253	ICMP	590 Destination unreachable (Port unreachable)

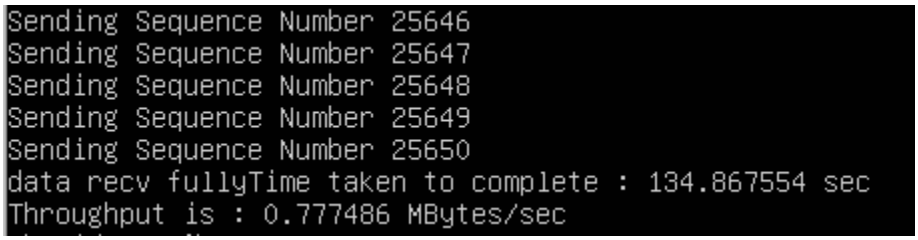
Frame 143124: 4138 bytes on wire (33104 bits), 4138 bytes captured (33104 bits) on interface bri0, id 0

Throughput calculated fromWireshark:

Throughput = ***File size / average effective bandwidth***
= 104857600 bytes / (135.197-0.0)sec
= 775591.17436 bytes/sec
= **0.775 MBps**

One of capture for File Transfer:

Server Side:



Client Side:

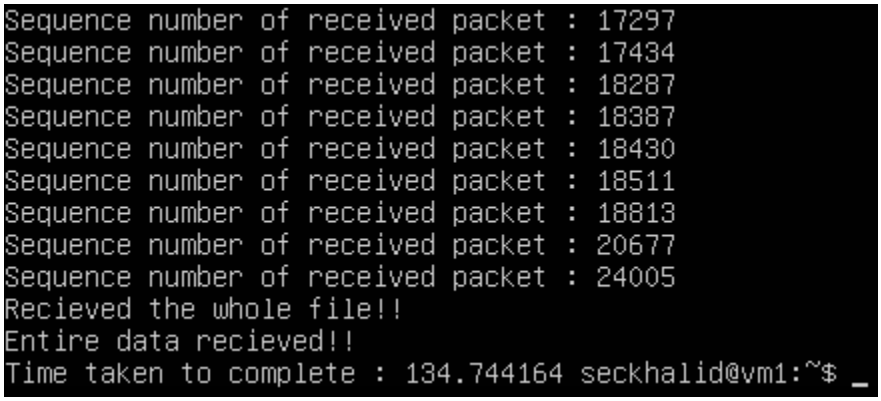
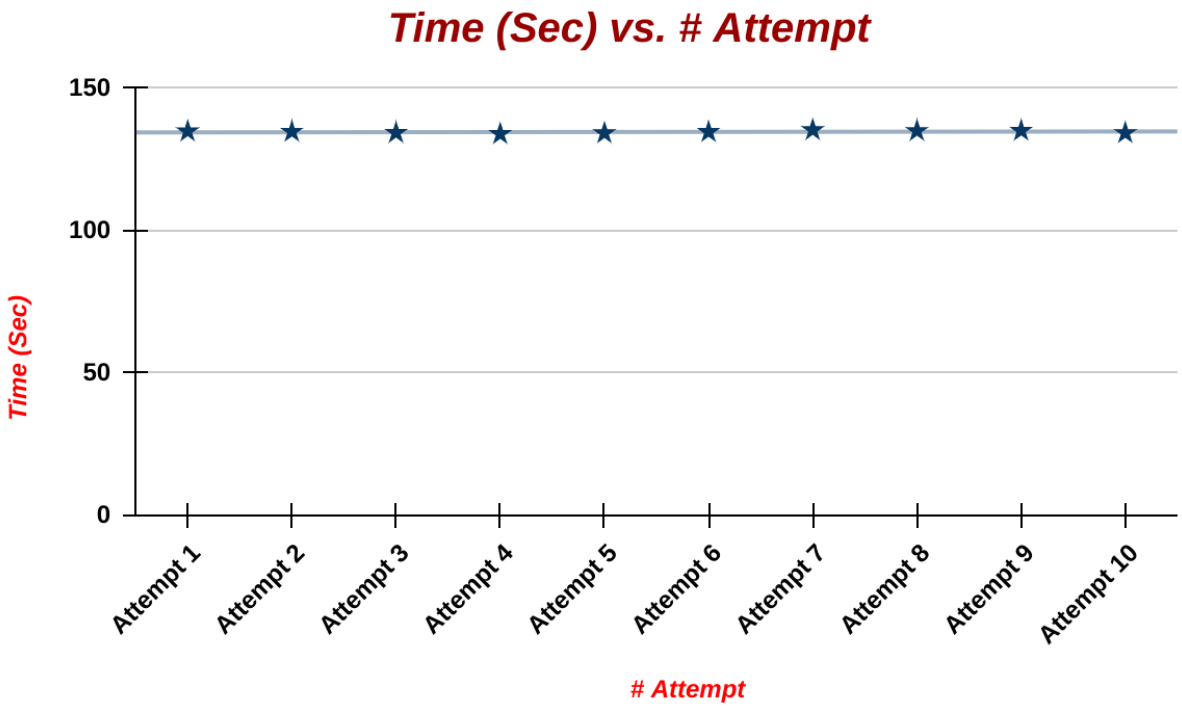
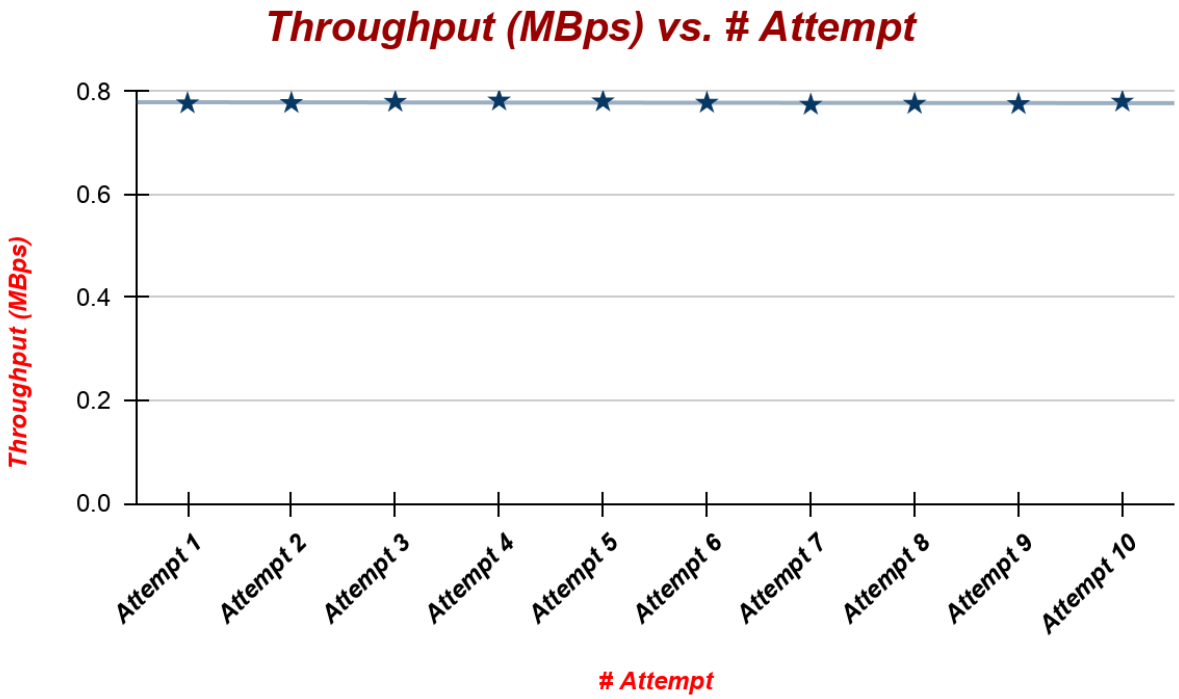


Table and Graphs representing transfer of 100MB file with 5% loss and 50ms delay over 100Mbit Link

With delay, packet loss	Time (Sec)	Throughput (MBps)
Attempt 1	134.867554	0.7774857398
Attempt 2	134.744164	0.778197711

Attempt 3	134.354106	0.7804569813
Attempt 4	133.944164	0.7828456042
Attempt 5	134.231498	0.7811698563
Attempt 6	134.678132	0.7785792574
Attempt 7	135.342564	0.7747570084
Attempt 8	135.030319	0.7765485617
Attempt 9	135.114369	0.7760654975
Attempt 10	134.292752	0.7808135468
Average	134.6599622	0.7786843119





We can see that the average throughput of the File transfer for 10 runs is close to that of measured through wireshark packet capture. **(0.775 ~ 0.778)**