CS F303 Computer Networks: 2020-21 semester II
Assignment 2
Performance Documentation

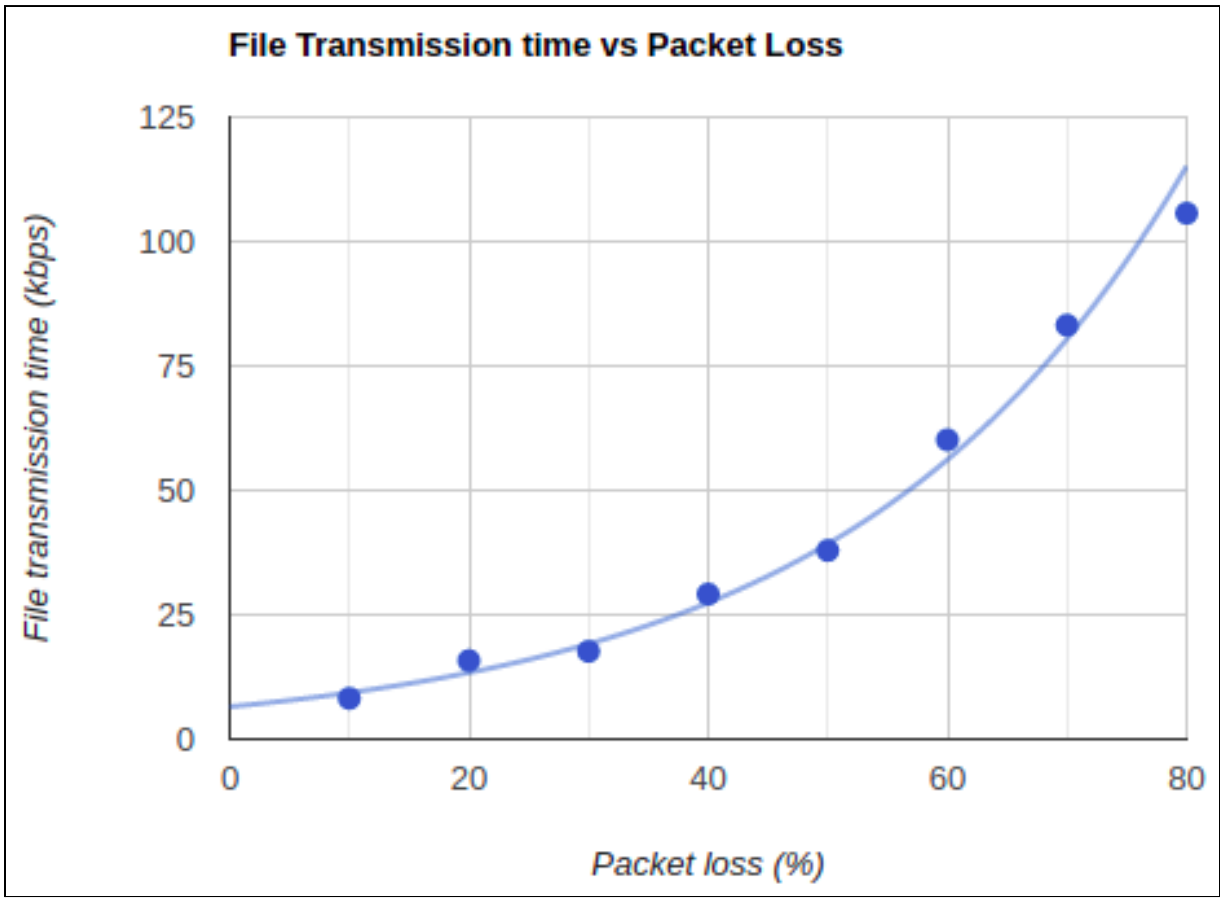
Team members:

1. Pranay Tarigopula 2018A7PS0237H
2. D Vishal Dheeraj 2018A7PS0239H
3. B Rishi Saimshu Reddy 2018A7PS0181H
4. Dhruv Adlakha 2018A7PS0303H
5. Abhinav Bandaru 2018A7PS0236H
6. P Pranav Reddy 2018A7PS0238H

Table Of Contents

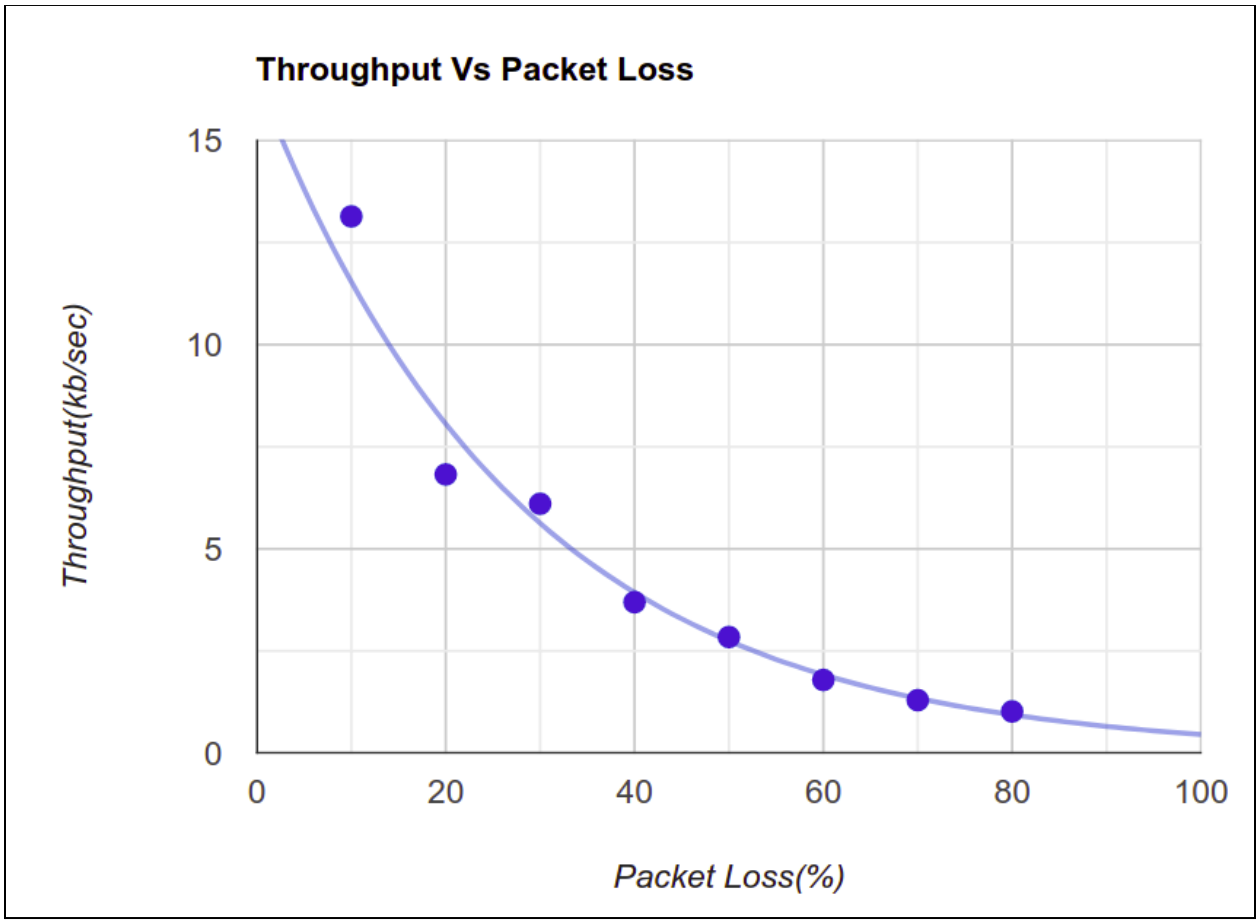
Performance against Packet Loss

Below is the performance of our implementation of a reliable UDP protocol against packet loss during transmission. We ran our implementation to send a file of 107.7kb against packet loss percentages of 10-80% in increments of 10%. Testing beyond 80% while keeping the other parameter constant becomes tricky as packets are lost more often than being sent. As the transmission time increased exponentially, the throughput decreased exponentially too.



FIle Transmission time (kbps)	Packet loss (%)
8.1990525	10
15.7915709	20
17.6488986	30
29.1548140	40
37.9181783	50
60.1451762	60

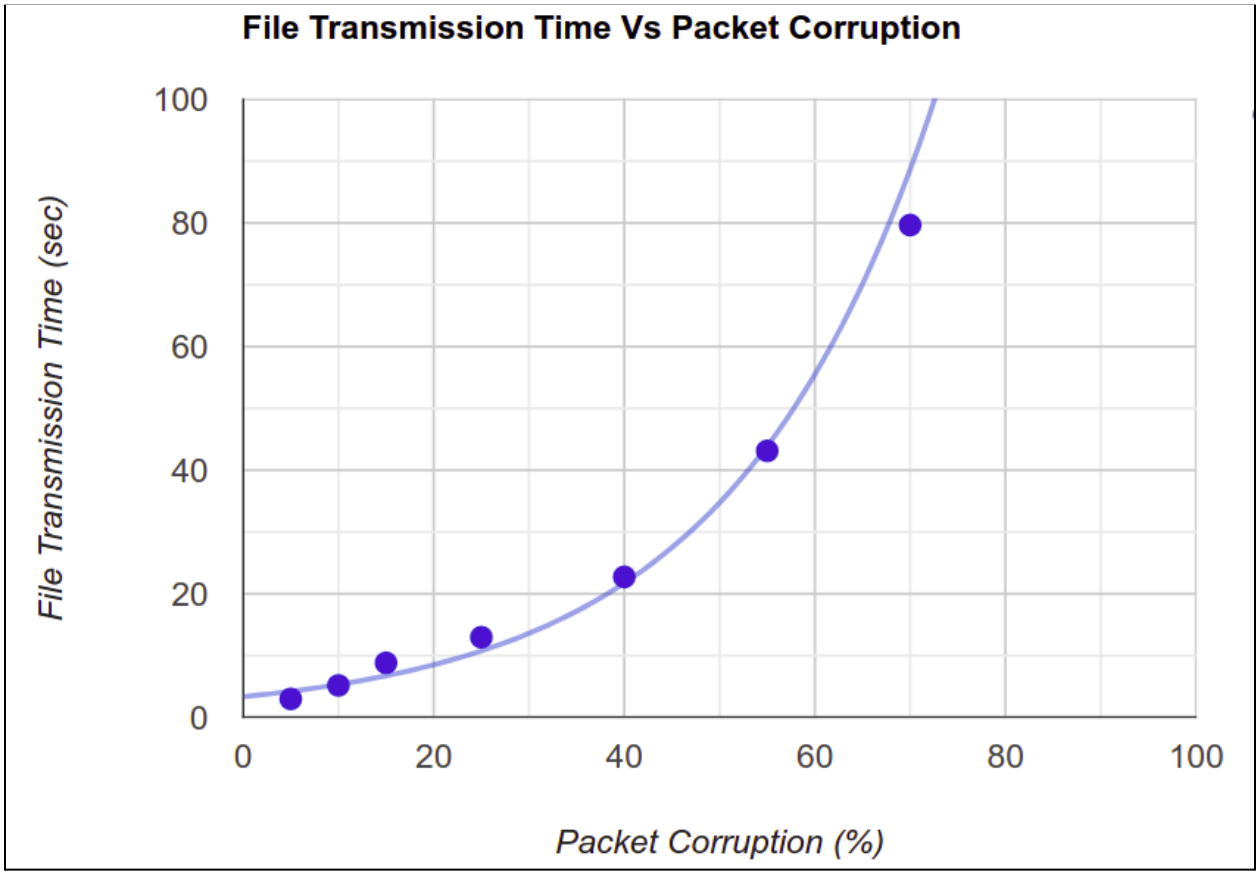
83.1892909	70
105.6869293	80



Throughput (kbps)	Packet Loss (%)
13.1378595270612	10
6.82123397869176	20
6.10338369783597	30
3.69469000899817	40
2.84080103078159	50
1.7906657131416	60
1.29485416734091	70
1.01921780406955	80

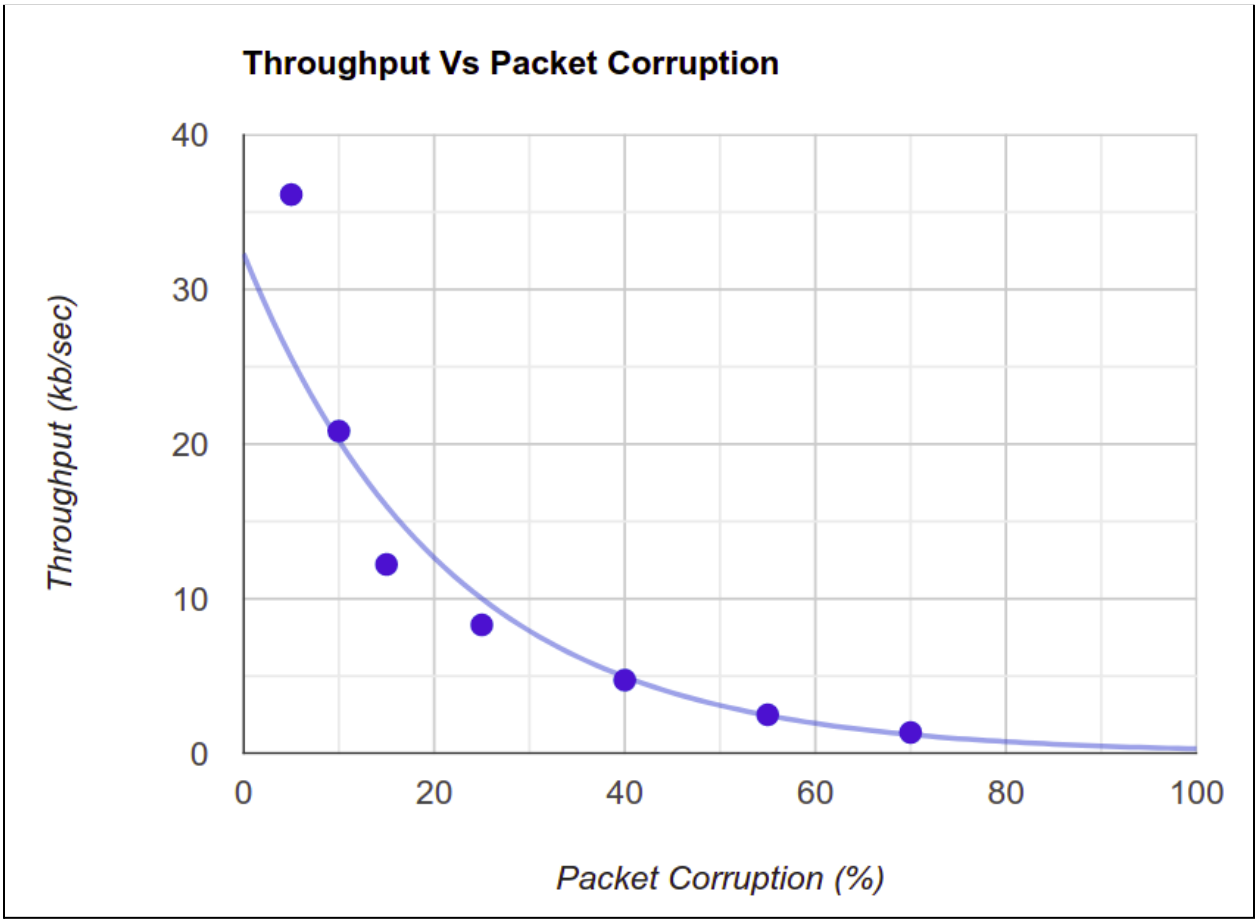
Performance against Packet Corruption

Below is the performance of our implementation of a reliable UDP protocol against packet corruption during transmission. We ran our implementation to send a file of 107.7kb against packet corruption percentages from 5% to 70%. Testing beyond 70% while keeping the other parameter constant becomes tricky as the retransmission limit was being reached often. This can be solved by increasing the limit but was kept unchanged for testing purposes. As the transmission time increases exponentially, the throughput decreases exponentially too.



File transmission time (in kbps)	Packet corruption(%)
2.981191	5
5.169463	10
8.816090	15
12.962989	25
22.730120	40

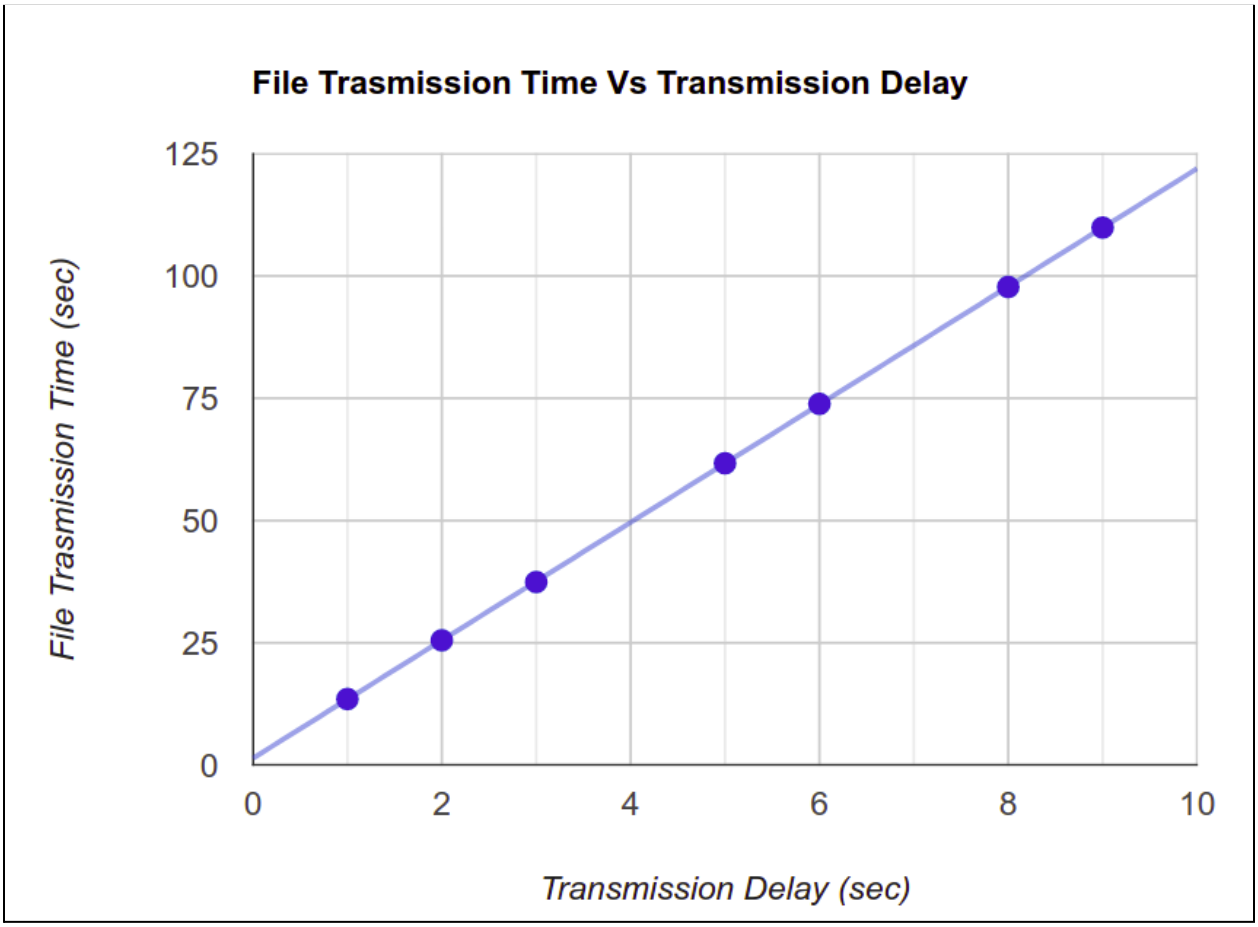
43.116407	55
79.649820	70



Throughput (kb/sec)	Packet corruption(%)
36.1325	5
20.8374	10
12.2183	15
8.3097	25
4.7390	40
2.4983	55
1.3524	70

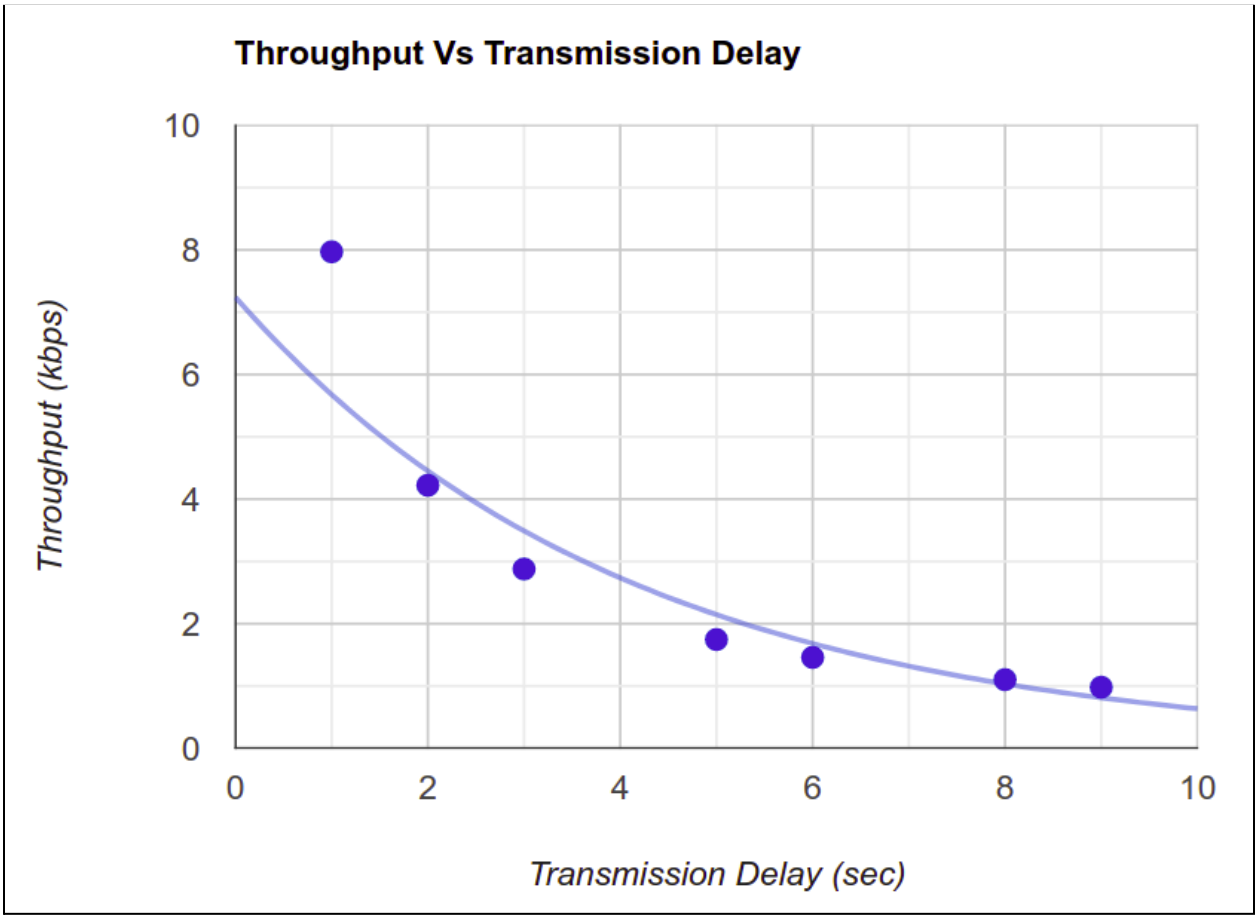
Performance against Transmission Delay

Below is the performance of our implementation of a reliable UDP protocol against delays during transmission. We ran our implementation to send a file of 107.7kb against transmission delays values from 1second to 9 seconds. Testing beyond 9 seconds is possible as long as timeout values are set accordingly albeit will take a lot of time to run as the delay is added to every packet transmission. As the transmission time increases exponentially, the throughput decreases exponentially too.



File Transmission Time (sec)	Transmission Delay (sec)
13.5168	1
25.5163	2
37.4279	3
61.7067	5
73.8520	6

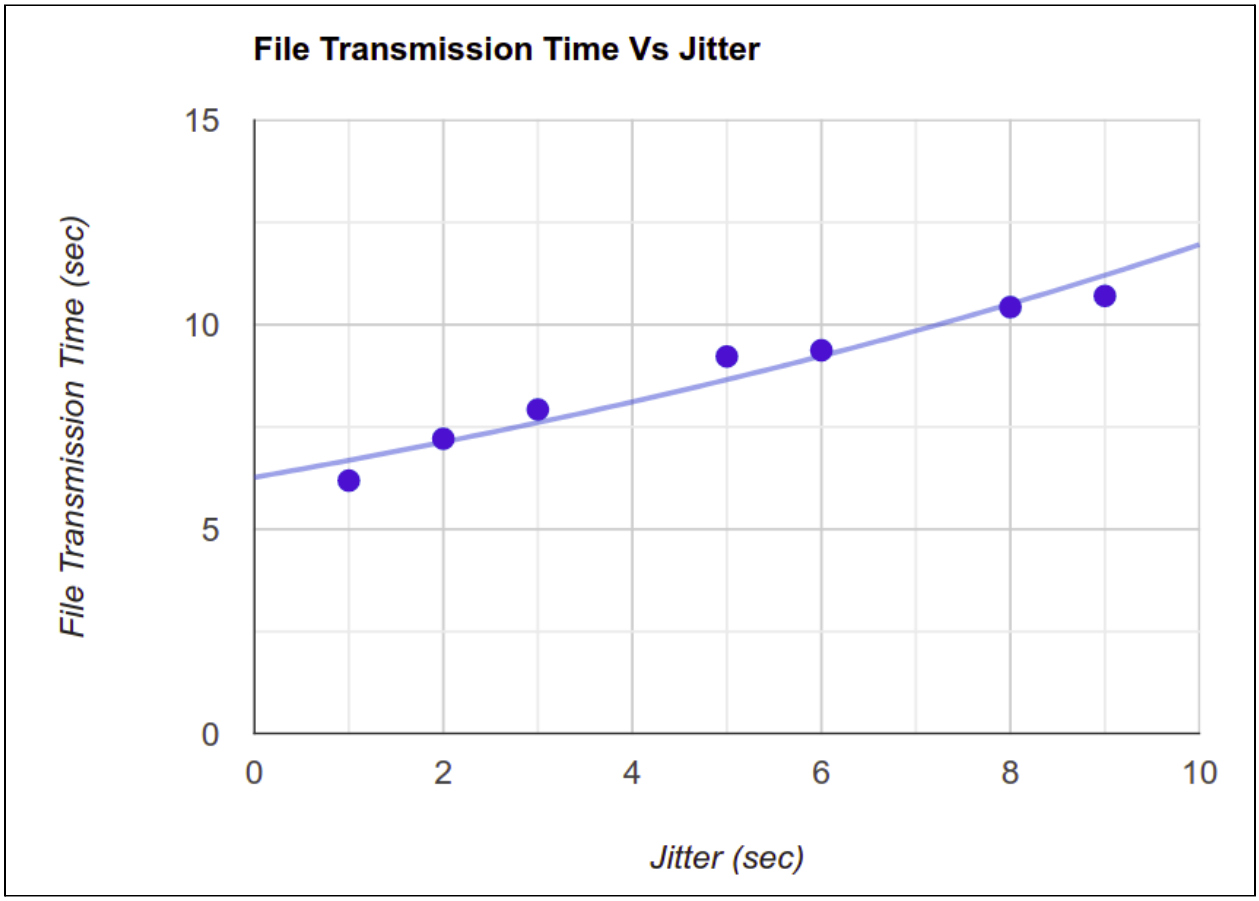
97.7504	8
109.8863	9



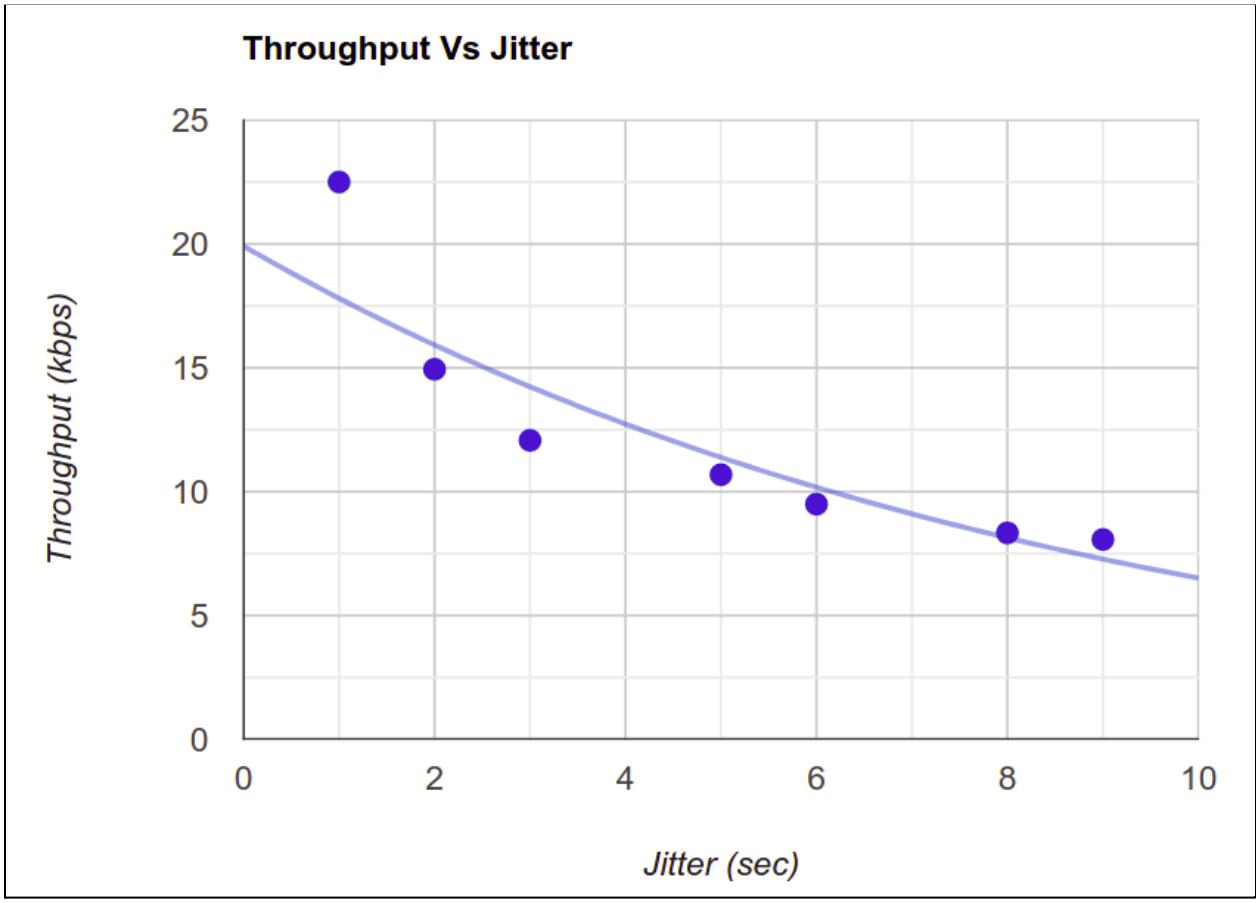
Throughput (kbps)	Transmission Delay (sec)
7.9692	1
4.2215	2
2.8780	3
1.7456	5
1.4586	6
1.1020	8
0.9803	9

Performance against Jitter

Below is the performance of our implementation of a reliable UDP protocol against Jitter during transmission. We ran our implementation to send a file of 107.7kb against Jitter values from 1 second to 9 seconds. As the transmission time increases exponentially, the throughput decreases exponentially too.



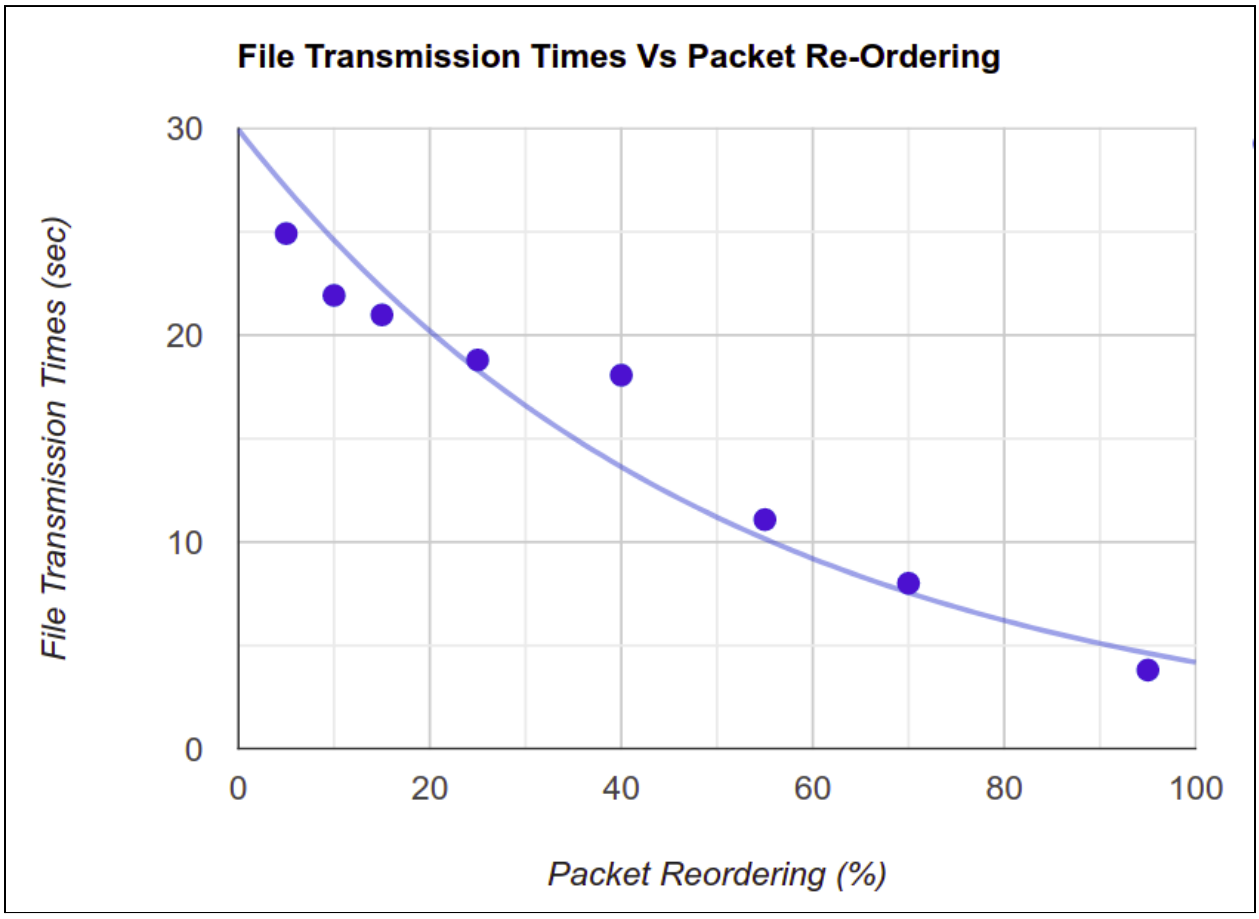
File Transmission Time (sec)	Jitter (sec)
6.1875	1
7.2110	2
7.9256	3
9.2214	5
9.3719	6
10.4282	8
10.7017	9



File Transmission Time (sec)	Jitter (sec)
22.4999	1
14.9380	2
12.0684	3
10.6814	5
9.4937	6
8.3295	8
8.0655	9

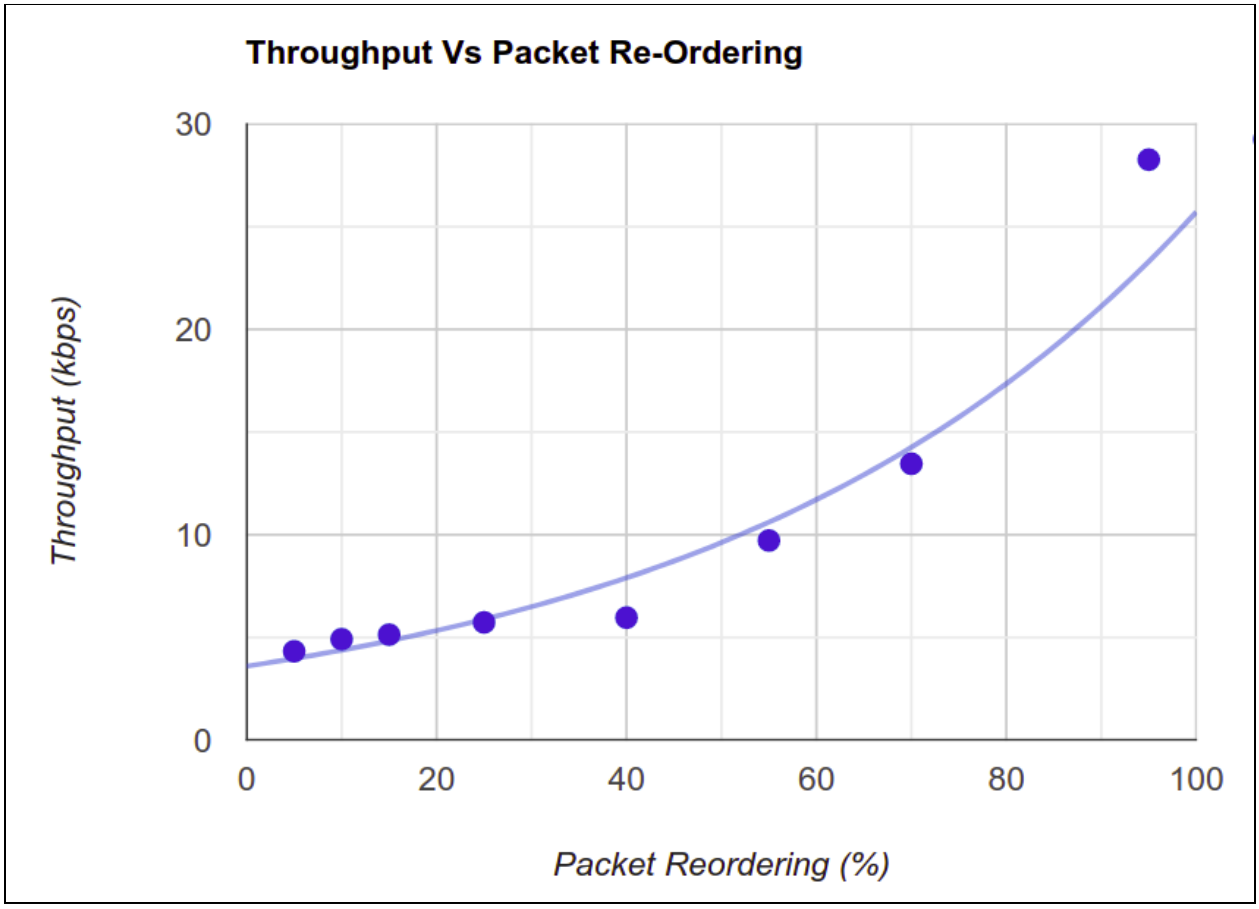
Performance against Packet Reordering

Below is the performance of our implementation of a reliable UDP protocol against packet reordering during transmission. We ran our implementation to send a file of 107.7kb against packet reordering percentages from 5-95%. This X percentage passed on to netem for emulation invokes a packet reordering after every x% of the packets sent, hence testing cannot be done for values of 0% and 100% since reordering doesn't happen for these values. As the transmission time decreased exponentially, the throughput increased exponentially too.



File Transmission Times (sec)	Packet Reordering (%)
24.9126668794961	5
21.9229401744997	10
20.9796565049983	15
18.7982409470023	25
18.0647123959971	40

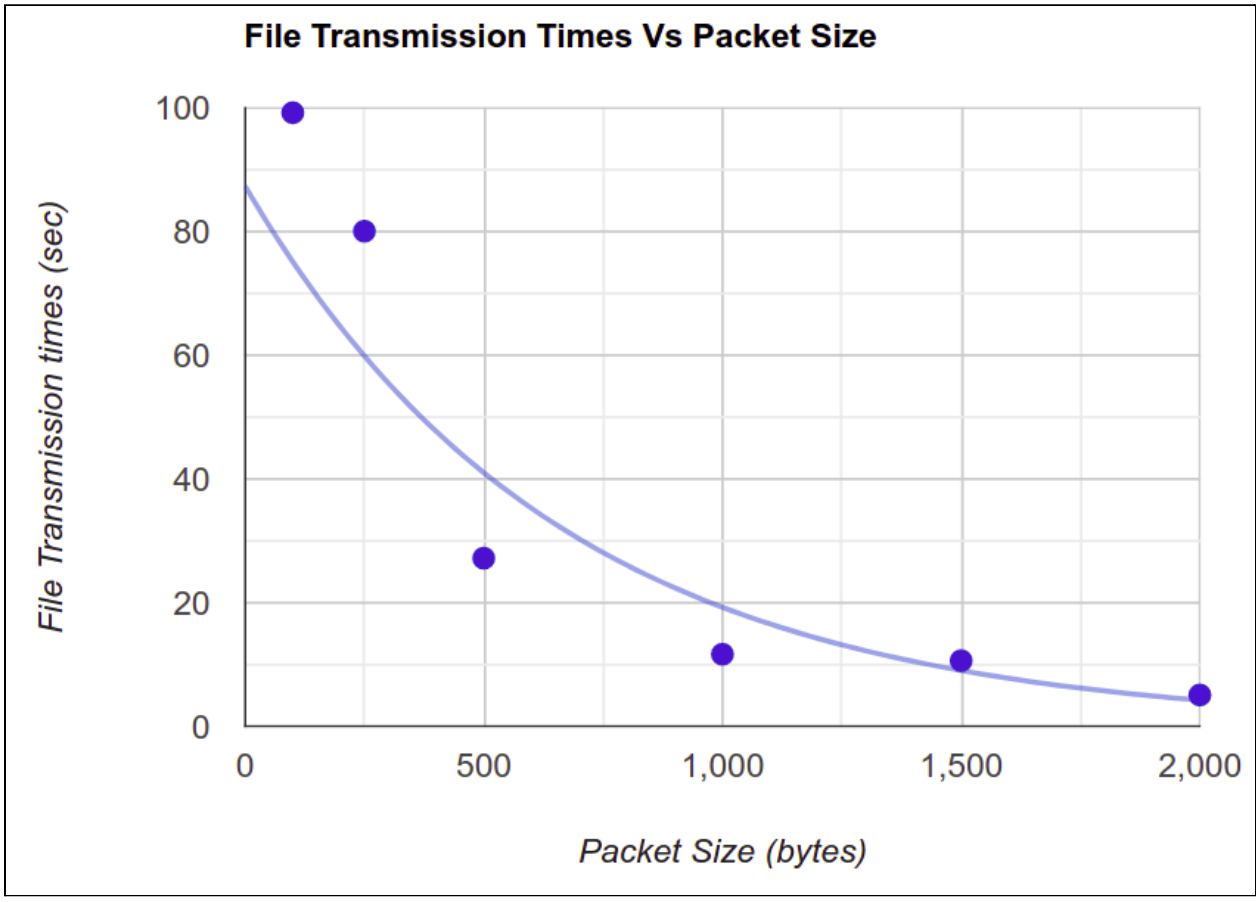
11.0842473164972	55
8.00630934549918	70
3.81141884450335	95



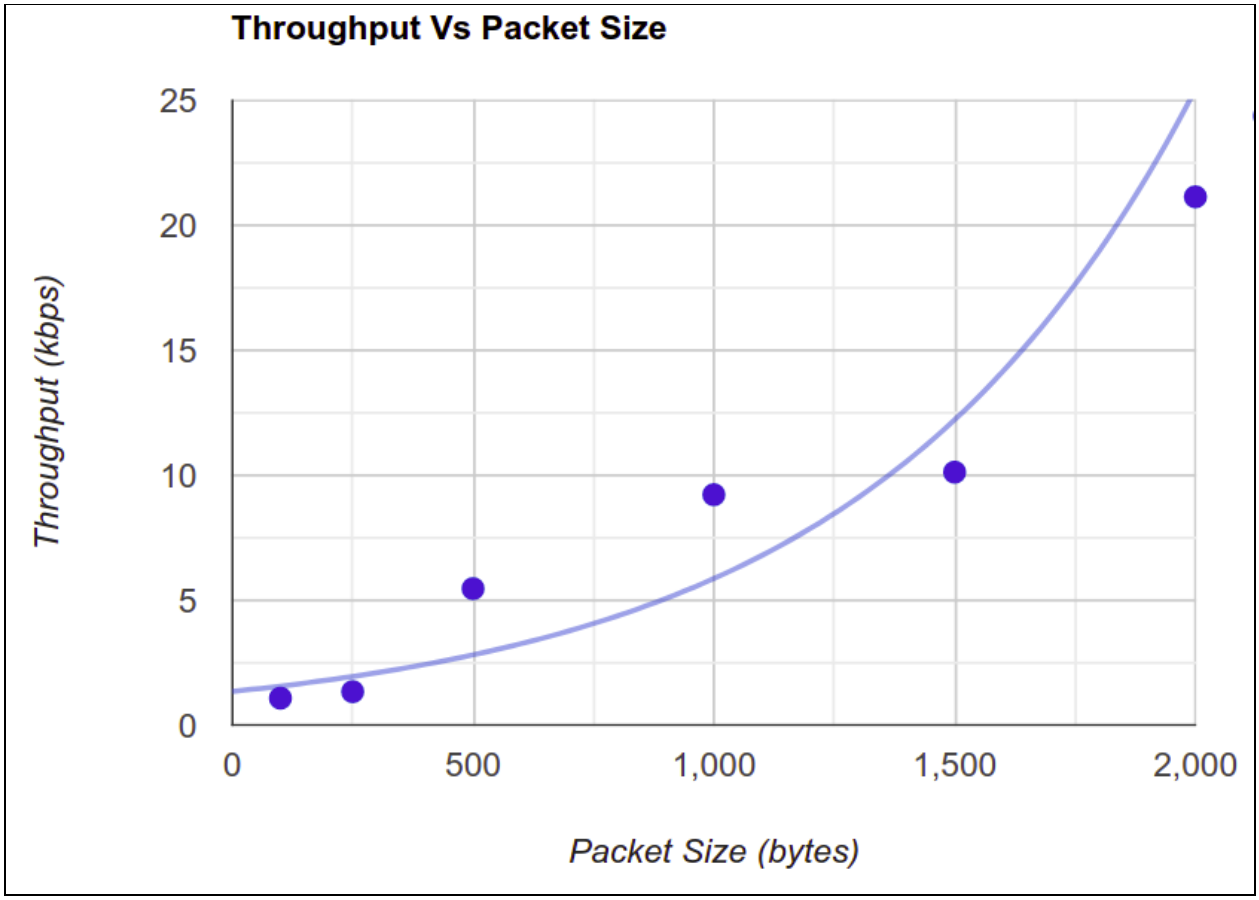
Throughput (kbps)	Packet Reordering (%)
4.32382452352603	5
4.91348328018955	10
5.13440246146722	15
5.73021700826626	25
5.96289592874278	40
9.7181158922427	55
13.4541391484648	70
28.2619162035539	95

Performance against Packet Size

Below is the performance of our implementation of a reliable UDP protocol against packet sizes during transmission. We ran our implementation to send a file of 107.7kb against packet reordering percentages from 100-2000 bytes. As packet sizes increase, the number of packets to be sent reduces which helps in reducing the transmission times and as the transmission time decreases exponentially, the throughput increases exponentially.



File transmission time (sec)	Packet Size (Bytes)
99.1990106145447	100
80.0426828705022	250
19.6994221479999	500
11.6765276422484	1000
10.6379394062478	1500
5.09540865324925	2000



Throughput (kbps)	Packet Size (Bytes)
1.08587776564181	100
1.34575699035816	250
5.46807917464404	500
9.22517406718169	1000
10.1258331981789	1500
21.1402082404735	2000