CS3543

Computer Networks - II

Assignment - I

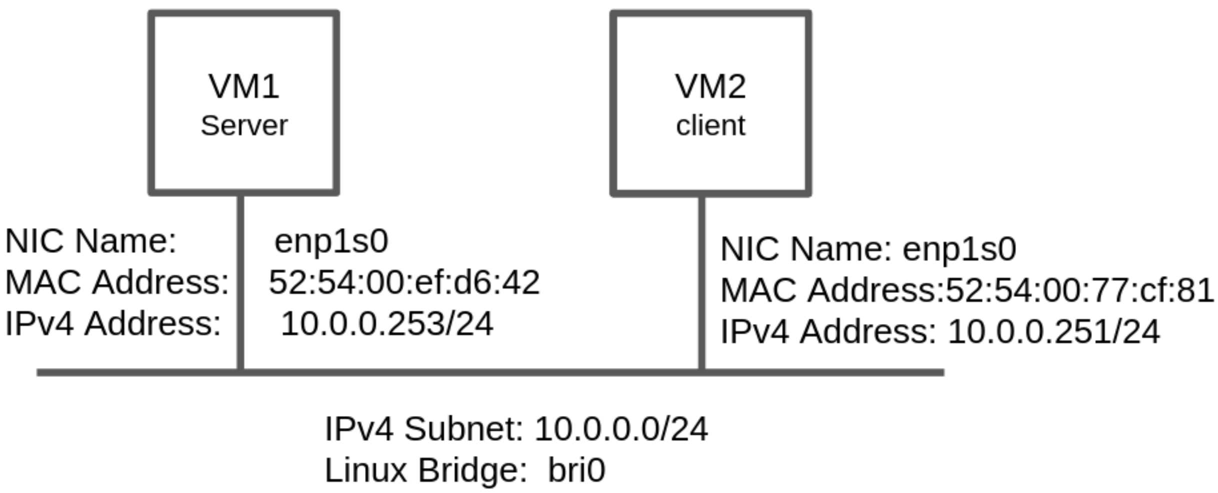
Implementing "Better RDT over UDP" than TCP

Team Members:

CS19BTECH11048 - GOKUL
ES19BTECH11003 - HEMANTH
CS19BTECH11020 - SHARANYA
CS19BTECH11003 - SARAT
CS19BTECH11009 - NAVEEN

Task -1

Network diagram:



Network diagram for PC 1

We performed 10 attempts of FTP using TCP protocol to transmit data (the given 100 MB file) from the one Ubuntu server VM (client) to another Ubuntu server VM (server) on two different computers and we report the overall throughput, and time taken to complete to transfer 100 MB data of each attempt in the below table for all 3 computers. We used the tc command to configure traffic control in Ubuntu. We performed 10 attempts under 2 situations - No delay, no packet loss, 100 mbps link and 50 ms delay, 5% packet loss, 100 mbps link. The time, throughput we recorded are as follows:

1) Without delay and loss:

PC 1:

```
sarat2@sarat2:~$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
sarat2@sarat2:~$ ftp 10.0.0.253
Connected to 10.0.0.253.
220 (vsFTPd 3.0.3)
Name (10.0.0.253:sarat2): ftpuser_s
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
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 8.55 secs (11.6916 MB/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 8.50 secs (11.7692 MB/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 8.56 secs (11.6796 MB/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 8.56 secs (11.6774 MB/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 8.53 secs (11.7171 MB/s)
ftp> _
```

PC 2:

```
hemanth2@server2-h:~$ sudo tc qdisc add dev enp1s0 root netem rate 100mbit
hemanth2@server2-h:~$ ftp 10.0.0.253
Connected to 10.0.0.253.
220 (vsFTPd 3.0.3)
Name (10.0.0.253:hemanth2): ftpuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 1000     1000     104857600 Feb 16 11:07 data
226 Directory send OK.
ftp> get data
local: data remote: data
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.80 secs (11.3666 MB/s)
ftp> get data
local: data remote: data
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.78 secs (11.3844 MB/s)
ftp> get data
local: data remote: data
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.80 secs (11.3680 MB/s)
ftp> get data
local: data remote: data
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.81 secs (11.3488 MB/s)
ftp> get data
local: data remote: data
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.77 secs (11.4087 MB/s)
ftp> _
```

Attempt	Time taken(secs)		Overall Throughput(MB/s)	
	PC 1	**PC 2**	**PC 1**	**PC 2**
1	8.55	8.80	11.695	11.3666
2	8.5	8.78	11.764	11.3844
3	8.56	8.80	11.682	11.3680
4	8.56	8.81	11.682	11.3488
5	8.53	8.87	11.723	11.4087
6	8.59	8.82	11.641	11.4023
7	8.52	8.79	11.737	11.3542
8	8.57	8.80	11.668	11.3342
9	8.56	8.81	11.682	11.4102
10	8.51	8.87	11.750	11.3622

Average Time taken for PC 1: 8.545 s
Average Throughput for PC 1: 11.7024 MB/s

Average Time taken for PC 2: 8.815 s
Average Throughput for PC 2: 11.374 MB/s

Throughput = Total Data Transferred/Total time taken

We can see that both PC's have almost the same Time taken and throughput. This is because we are using the exact same protocol in both. The small variations are due to randomness in TCP protocol.

2) With 50ms delay and 5% packet loss:

PC 1:

```
sarat2@sarat2:~$ sudo tc qdisc add dev enp1s0 root netem delay 50ms loss 5% rate 100Mbit
sarat2@sarat2:~$ ftp 10.0.0.253
Connected to 10.0.0.253.
220 (vsFTPd 3.0.3)
Name (10.0.0.253:sarat2): ftpuser_s
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
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 1292.95 secs (79.1990 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 1309.72 secs (78.1846 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 1324.03 secs (77.3397 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 1365.52 secs (74.9898 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 1294.77 secs (79.0876 kB/s)
ftp> _
```

PC 2:

```
hemanth2@server2-h:~$ sudo tc qdisc add dev enp1s0 root netem delay 50ms loss 5% rate 100mbit
[sudo] password for hemanth2:
hemanth2@server2-h:~$ ftp 10.0.0.253
Connected to 10.0.0.253.
220 (vsFTPd 3.0.3)
Name (10.0.0.253:hemanth2): ftpuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
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 1294.49 secs (79.1047 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 1322.60 secs (77.4230 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 1333.55 secs (76.7876 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 1299.46 secs (78.8021 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 1293.68 secs (79.1541 kB/s)
ftp> _
```

Attempt	Time taken(secs)		Overall Throughput(KB/s)	
	PC1	**PC 2**	**PC 1**	**PC 2**
1	1292.95	1294.49	77.342	77.2505
2	1309.72	1322.60	76.352	75.6086
3	1324.03	1333.55	75.526	74.9878
4	1365.52	1299.46	73.232	76.9551
5	1294.77	1293.68	77.233	77.2988
6	1333.66	1296.57	74.981	77.1265
7	1327.52	1328.77	75.328	75.2575
8	1297.56	1333.66	77.067	74.9816
9	1294.36	1298.63	77.258	77.0042
10	1366.72	1296.58	73.167	77.1259

Average Time taken for PC 1: 1320.681 s
Average Throughput for PC 1: 75.7486 KB/s

Average Time taken for PC 2: 1309.799 s
Average Throughput for PC 2: 76.359 KB/s

We can see that both PC's have almost the same Time taken and throughput. This is because we are using the exact same protocol in both, same delay/packet loss for the bridge in both the systems. The small variations are due to randomness in TCP protocol.

We can also see that as delay/packet loss increases, the time taken for a file to be transferred in FTP increases a lot. And, as time taken increases, the throughput decreases due to the packet loss and delay.

Task 2:

“Our udp-ftp” supports the following RDT features:

1) Packet loss detection
2) Acknowledgment
3) Packet corruption detection (using Checksum)
4) Packet retransmission
5) Flow control

Reliable data transfer over UDP:

Implementation Details :

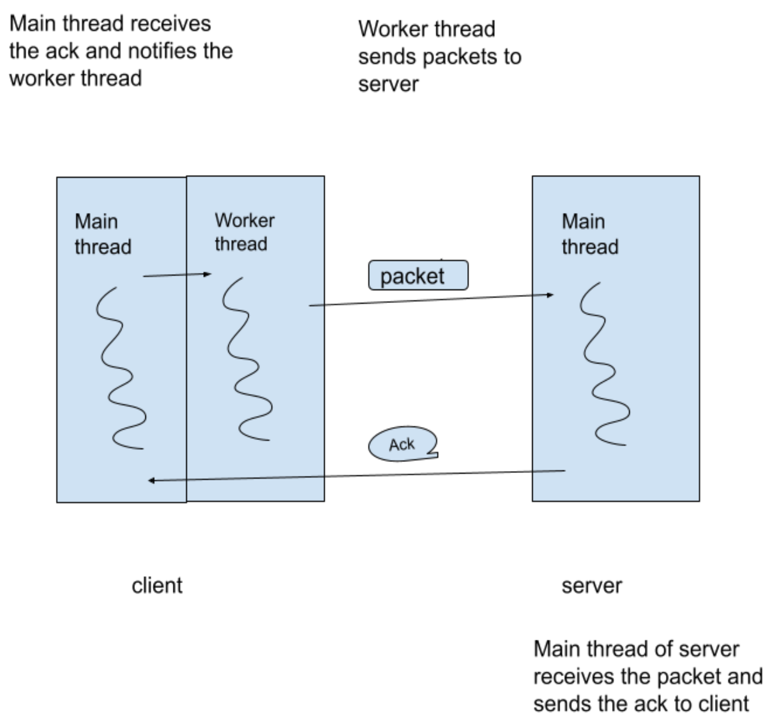
- We have made use of multiple threads to handle different operations in server and client codes
- In the client we have used 2 threads, one to handle the data transmission(to the receiver) and another thread to handle the acknowledgement that has been sent by the server.
- On the server we have only one main thread which handles sending acknowledgement and receiving packets.
- We have used **BOOL LIST** (of size = total no.of packets) to maintain the acknowledgement status that the server sends to the client. We update the list after receiving acknowledgement of the packet Id from the server.

We have implemented a reliable data transfer (RDT) protocol using UDP that provides features like packet loss detection, packet corruption detection, acknowledgement, packet retransmission, checksum, congestion and flow control.

- **Packet Loss Detection :** Our-UDP-FTP provides packet loss detection. When a packet is sent by a thread from the client side, the thread waits for acknowledgement from the server side. If it doesn’t receive acknowledgement from the server side within a fixed time, the thread will resend the packet assuming that the packet has been lost and didn’t arrive at the server.

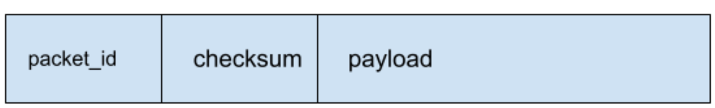
- **Packet Corruption Detection:** Server computes checksum for received data and sends acknowledgement only if the checksum matches with the checksum bits of the received data. So If the packet is corrupted the client doesn't receive acknowledgement and it retransmits the packet.
- **CheckSum:** At the client side, we will calculate the checksum for the data. We appended checksum bits at the client side while sending the packet to the server.
- **Acknowledgement :** On receiving a packet, the thread on the server side sends its acknowledgement by transmitting the packet ID and the ack list is updated on the client side.
- **Packet Retransmission :** On receiving the acknowledgement from the server, the client thread updates the ack list and retransmits the packets which did not receive acknowledgement.
- **Flow Control**: 1st packet sent by the thread contains the total no.of packets that the client wants to send to the server. After receiving this packet, the server dynamically allocates the memory for the buffer of size total no.of packets. As the server has a buffer of the required size, Flow control has been addressed.

Pictorial representation of thread working in client and server side



- It stores data of every packet in the buffer until all the packets are received. And finally, it writes the data to the output file

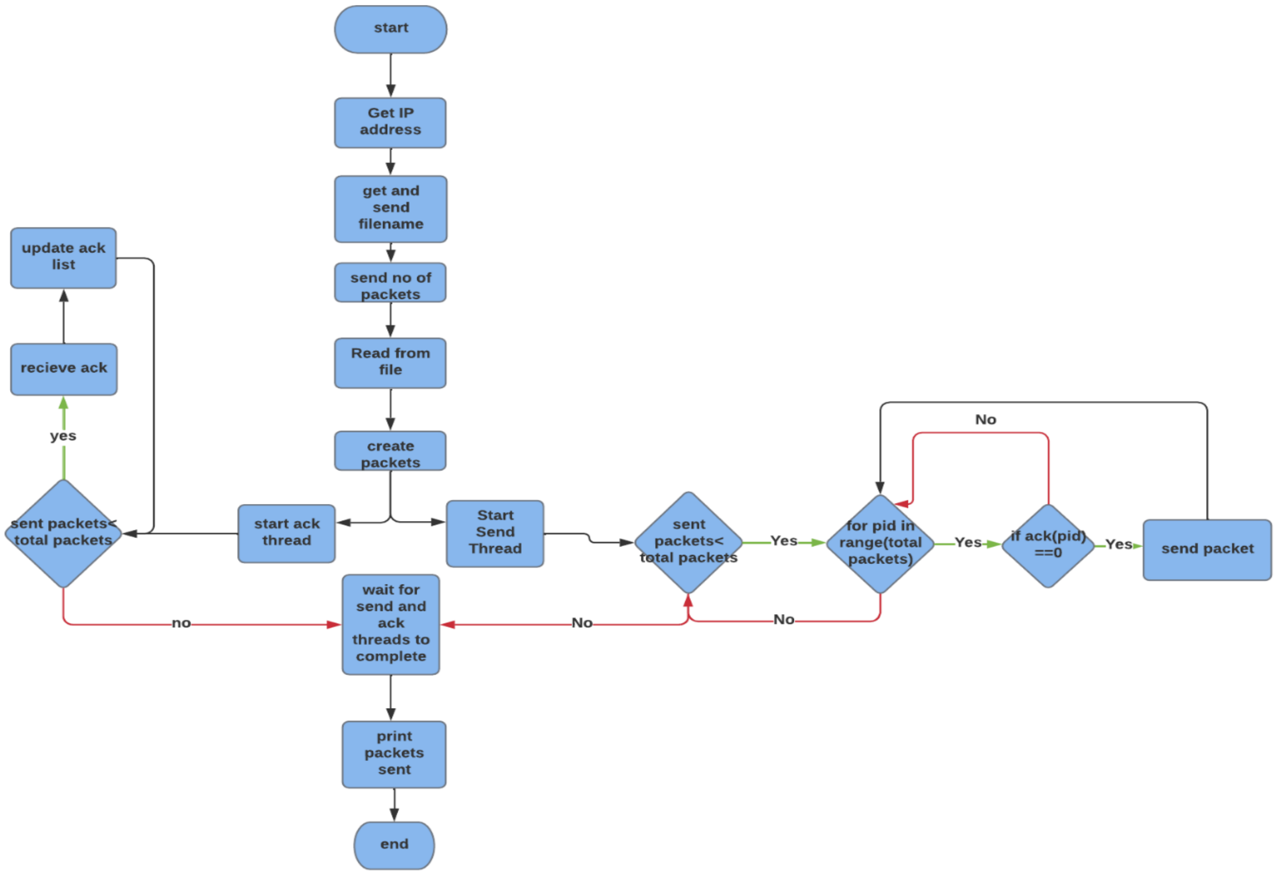
Application header:



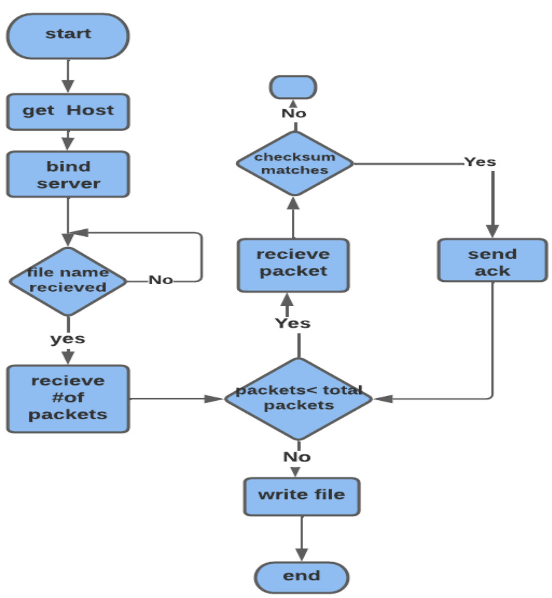
0-4 bytes	Packet ID
5-36 bytes	Checksum
37-8192 bytes	Payload

Flow Charts illustration of client and server process

CLIENT:



SERVER:



1) <u>Without Delay and Loss:</u>

- First, we will set the link speed to 100Mbps in both server and client using the “tc” command.
- `$ sudo tc qdisc add dev eth0 root netem rate 100Mbit`
- First we run the client.py in the client (`python3 client.py`). Then we run the server.py file in the server (`python3 server.py`).

```
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 10.890411616001074 s
Throughput:  9402.77 kB/s
sarat@sarat:~/CN2_A1/final$ python3 server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 10.579079796003498 s
Throughput:  9679.48 kB/s
sarat@sarat:~/CN2_A1/final$ python3 server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 10.6335315489996 s
Throughput:  9629.91 kB/s
sarat@sarat:~/CN2_A1/final$ python3 server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 10.591628972000763 s
Throughput:  9668.01 kB/s
sarat@sarat:~/CN2_A1/final$ python3 server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 10.596187612001813 s
Throughput:  9663.85 kB/s
sarat@sarat:~/CN2_A1/final$ _
```

Attempt	Time taken(sec)	Throughput (KB/s)
1	10.890	9402.77
2	10.579	9679.48
3	10.633	9629.91
4	10.591	9668.01
5	10.596	9663.85
6	10.596	9437.52
7	10.582	9450.00
8	10.634	9403.79
9	10.597	9436.63
10	10.590	9442.87

The average time taken = 10.628 s
The average throughput = 9521.48 KB/s

Wireshark

We installed wireshark in the host OS Ubuntu, and monitored the bridge we created (that connects the virtual machines)

No.	Time	Source	Destination	Protocol	Length	Info
362	12.041317962	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
367	12.042011884	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
369	12.042087273	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
375	12.042847803	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
376	12.042861505	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
382	12.043629524	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
383	12.043675520	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
389	12.044412553	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
390	12.044493530	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
396	12.045194578	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
397	12.045317068	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
403	12.045984885	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
404	12.046139946	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
410	12.046744462	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
412	12.047001512	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
417	12.047519064	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
419	12.047853930	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
424	12.048293767	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
427	12.048699762	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
431	12.049068014	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
435	12.049535480	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
438	12.049842806	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
442	12.050367754	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
445	12.050616424	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
450	12.051219454	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
452	12.051391115	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
458	12.052096721	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
459	12.052167944	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
465	12.052954197	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
466	12.053100468	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
472	12.053733546	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
474	12.053994140	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
479	12.054525713	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
482	12.054893310	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
486	12.055278624	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
489	12.055730341	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
493	12.056051664	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
498	12.056682592	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
500	12.056843026	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
505	12.057589341	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
507	12.057661510	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
513	12.058434213	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
514	12.058505417	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
520	12.059219089	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
521	12.059340672	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
527	12.060016329	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
528	12.060166312	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
534	12.060776217	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
537	12.061172860	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
541	12.061562223	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
545	12.062177054	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
548	12.062384032	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
553	12.063052571	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
555	12.063161530	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
561	12.063951856	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
562	12.063953558	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
568	12.064730638	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192
569	12.064888627	10.0.0.253	10.0.0.251	UDP	44	9999 → 58335 Len=2
575	12.065504824	10.0.0.251	10.0.0.253	UDP	834	58335 → 9999 Len=8192

Justification:

We can see that the time taken for file transfer is almost the same for TCP and UDP in case of no delay and no packet loss.
We can also see that there is only little difference between the time shown in the server and in wireshark, Hence it is consistent.

```
sarat@sarat:~/CN2_A1/final$ diff CS3543_100MB output
sarat@sarat:~/CN2_A1/final$ _
```

➢ We use diff command to verify if the file is correctly transferred or not. No output implies file has transferred correctly

2) With 50ms delay and 5% packet loss:

- First, we will change the rule in both client and server and add a 50 ms delay and 5% packet loss to the interfaces using the "tc" command.
- `$ sudo tc qdisc change dev eth0 root netem delay 50ms loss 5%`
- Now, we first run the client.py in the client (`python3 client.py`). Then we run the server.py file in the server (`python3 server.py`)

```
sarat@sarat:~/CN2_A1/final$ sudo tc qdisc add dev enp1s0 root netem delay 50ms loss 5% rate 100Mbit
sarat@sarat:~/CN2_A1/final$ python3  server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 402.8899630329979 s
Throughput:  254.16 kB/s
sarat@sarat:~/CN2_A1/final$ python3  server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 400.41539776199716 s
Throughput:  255.73 kB/s
sarat@sarat:~/CN2_A1/final$ python3  server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 402.94626462299857 s
Throughput:  254.13 kB/s
sarat@sarat:~/CN2_A1/final$ python3  server.py
Enter IP address of server: 10.0.0.253
Server starting in 10.0.0.253...
CS3543_100MB
Receiving File: CS3543_100MB
File Downloaded
No. of packets received: 12859
No. of bytes received: 104857600
Time Taken: 403.30501483600165 s
Throughput:  253.90 kB/s
sarat@sarat:~/CN2_A1/final$ _
```

The above image shows the transferring of file - CS3543_100MB

Attempt	Time taken(sec)	Throughput (KB/s)
1	402.88	248.212
2	400.41	249.744
3	402.94	248.175
4	402.95	248.169
5	403.30	247.954
6	401.92	248.805
7	400.52	249.675
8	402.69	248.329
9	403.57	247.788
10	402.28	248.583

The average time taken = 402.34 s
The average throughput = 248.54 KB/s

We can clearly see that our R-UDP application is performing better than the TCP when there is a delay and packet loss in the link. The increase in the time taken in case of delay and packet loss for our reliable UDP application is not as high as that of TCP. And therefore the throughput in case of our reliable UDP application is much better than that of TCP.

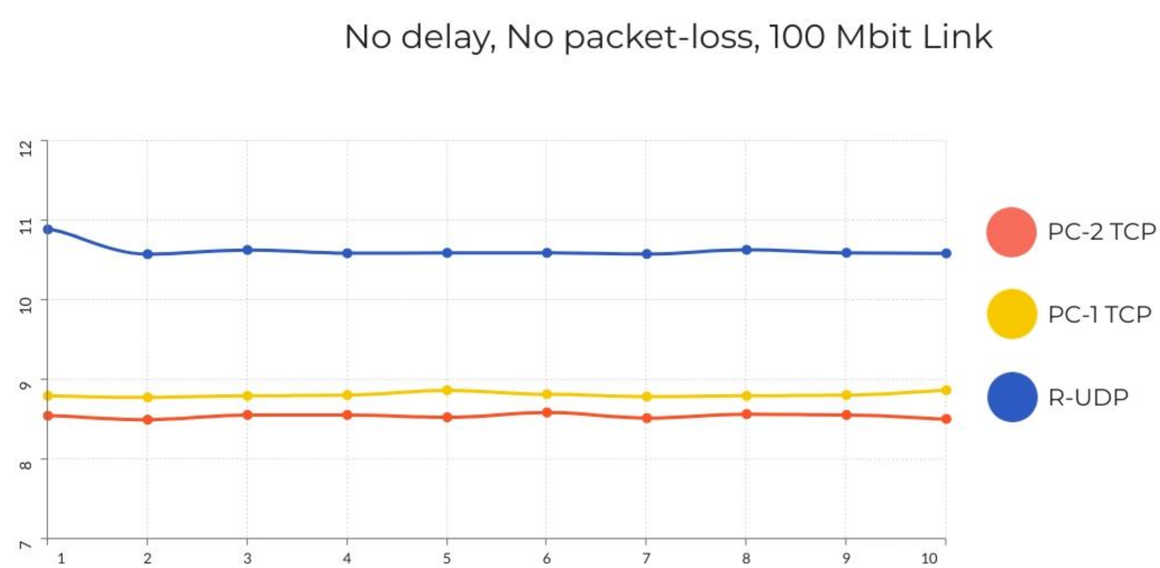
No.	Time	Source	Destination	Protocol	Length	Info
2935…	402.128385299	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.129458406	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.130221928	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.130919501	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.132512503	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.133734721	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.136333663	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.136910552	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.137674991	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.138440056	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.138961846	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.139848242	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.140951310	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.141257643	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.143023980	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.143974134	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.144670363	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.145836264	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.147028055	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.147483203	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.148125857	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.149018246	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.149999195	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.151548824	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.151893880	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.153089159	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.153735457	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.155402590	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.156164181	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.156376452	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.157258382	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.158460148	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.158802807	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.159437335	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.160755681	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.161081391	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.161969991	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.163376402	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.164018050	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.164785268	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.166094635	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.166820768	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.167598560	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.167806761	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.168700421	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.169477316	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.170364933	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.172657839	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.173728865	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192
2935…	402.173939700	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.174825676	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.175836320	10.0.0.251	10.0.0.253	UDP	1514	60860 → 9999 Len=8192
2935…	402.176781115	10.0.0.251	10.0.0.253	UDP	834	60860 → 9999 Len=8192

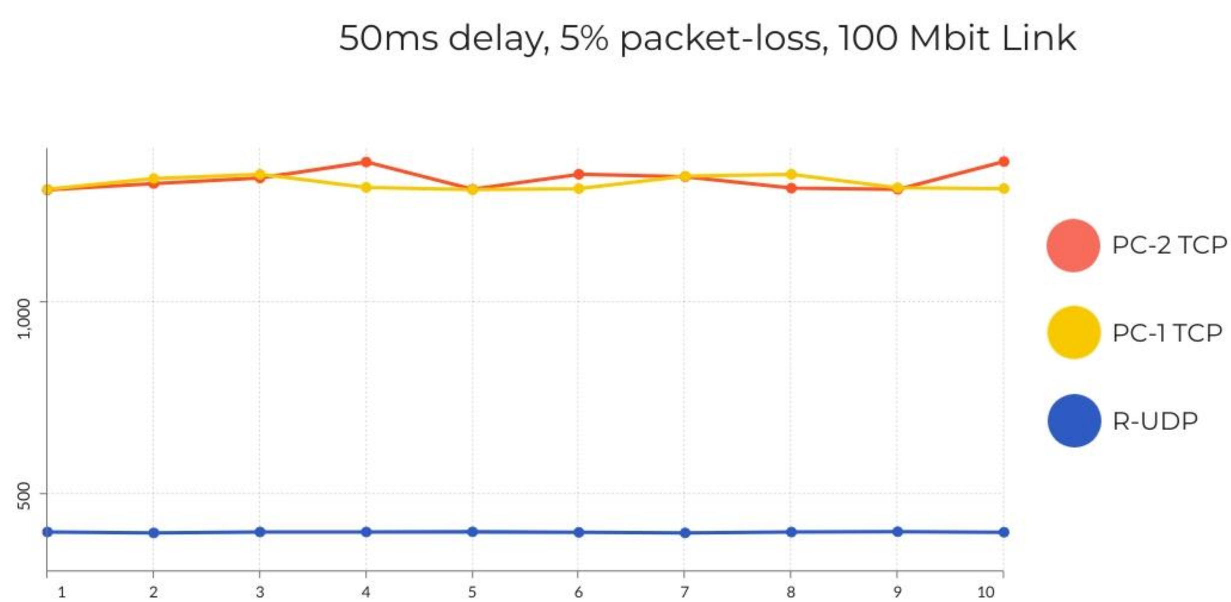
Justification:

```
sarat@sarat:~/CN2_A1/final$ diff CS3543_100MB output
sarat@sarat:~/CN2_A1/final$ _
```

- We use diff command to verify if the file is correctly transferred or not. No output implies file has transferred correctly
- We can also see that the values in wireshark are almost the same as the values we got in the server.
- We can see that our Reliable UDP application is 3.4 times faster than TCP in this case.

- This is because, in a somewhat reliable network (like the bridge we used), UDP is faster than TCP, because of lower packet overheads, and lower number of packets.





Summary:

- We are submitting the files - server.py, client.py along with this report.
- Run server.py on a computer and give the server IP as the IP address, when asked for it.
- Run client.py on another computer and give the server IP as the IP address of the server, when asked for it.
- Enter the file name of the file you want to send from client to server, and press Enter.
- After the transfer is done, you will be prompted with a message saying the transfer is done on the client side. On the server side, the time taken for the transfer is displayed along with a message saying the transfer is completed.

Future Improvements:

- We can optimize our code, to give even higher throughput.
- One of the ideas is to use multiple threads to send packets instead of the single one we have used now.
- We can come up with a better interface to our application.
- Instead of single transfer and closing the session, we can try to keep the server running forever, and a client can connect and send data at any time.