Computer Networks

Assignment 1

K.RITHVIK -CS19BTECH11038

A.MAHIDHAR-CS19BTECH11046

P.SRIVARDHAN-CS19BTECH11052

V.NIKHILISH-ES19BTECH11030

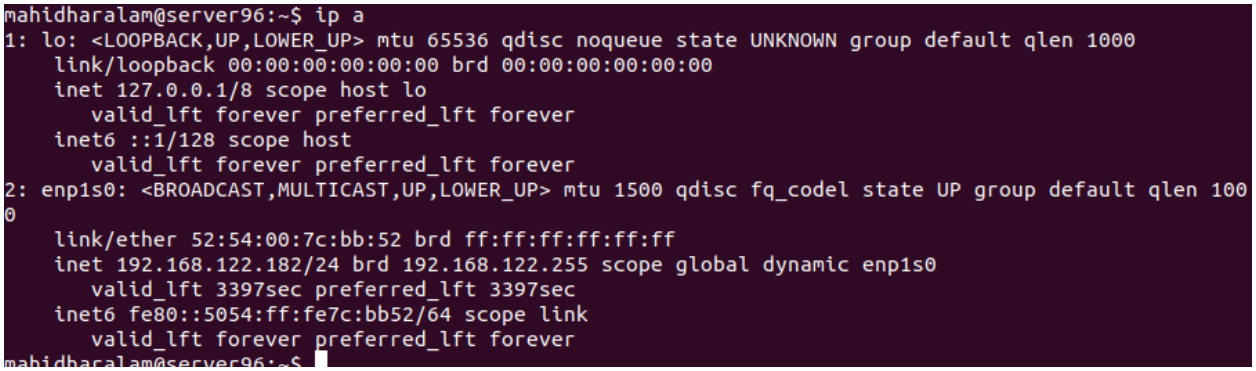
M.SHATHANAND-CS19BTECH11005

TASK 1:

Implement the testbed which contains 2 directly connected Linux VMs with 100Mbps link speed (must be configured on Linux VM), and apply the delay and packet loss using "tc" command.

VM1 : ubuntu20.04

IP Address:192.168.122.182/24



Vm2: ubuntu20.04-clone

IP Address: 192.168.122.237

```
mahidharalam@server96:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:8d:6c:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.237/24 brd 192.168.122.255 scope global dynamic enp1s0
       valid_lft 3386sec preferred_lft 3386sec
    inet6 fe80::5054:ff:fe8d:6c22/64 scope link
       valid_lft forever preferred_lft forever
```

PART 1:

1) **RUN the command `sudo tc qdisc add dev enp1s0 root netem rate 100Mbit` in VMs**

```
mahidharalam@server96:~$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
[sudo] password for mahidharalam:
```

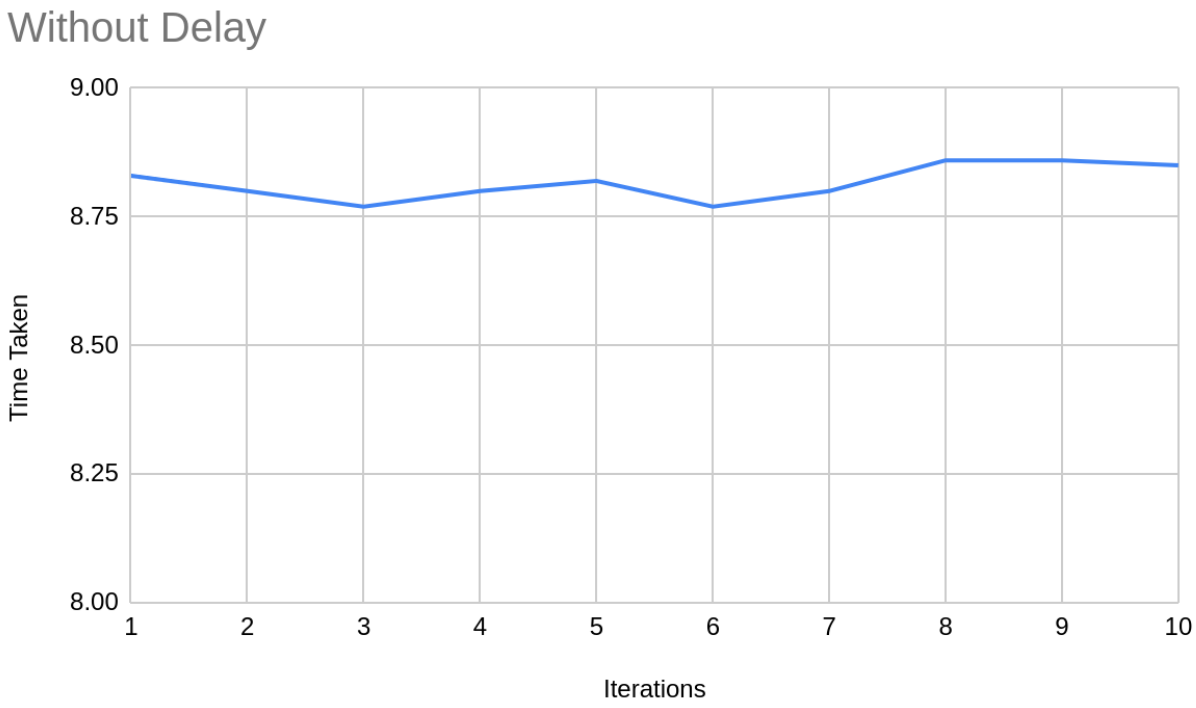
2) **Establish FTP connection in VM2 to VM1 for file transfer**

```
mahidharalam@server96:~$ ftp 192.168.122.182
Connected to 192.168.122.182.
220 (vsFTPd 3.0.3)
Name (192.168.122.182:mahidharalam): mahidharalam
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

3) **Recv the file from VM2 using `recv filename` command for 10 times**

```
ftp> recv CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for CS3543_100MB (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 8.83 secs (11.3314 MB/s)
```

4) **Average Time Taken is 8.81s**



5) Throughput calculation :

- **a) Average Effective Bandwidth/Throughput taken is 11.35 MB/s which is 90.8Mbps**
- **b) File size is 104857600bytes==104857600*8bits**
- **c) Time =(FIlesize)/Bandwidth)**
  - **i) 104857600*8/(90.8*1024*1024)=8.81s**

PART 2:

1) **RUN the command sudo tc qdisc change dev enp1s0 root netem delay 50ms loss 5% in VMs**

```
mahidharalam@server96:~$ sudo tc qdisc change dev enp1s0 root netem delay 50ms loss 5%
[sudo] password for mahidharalam:
```

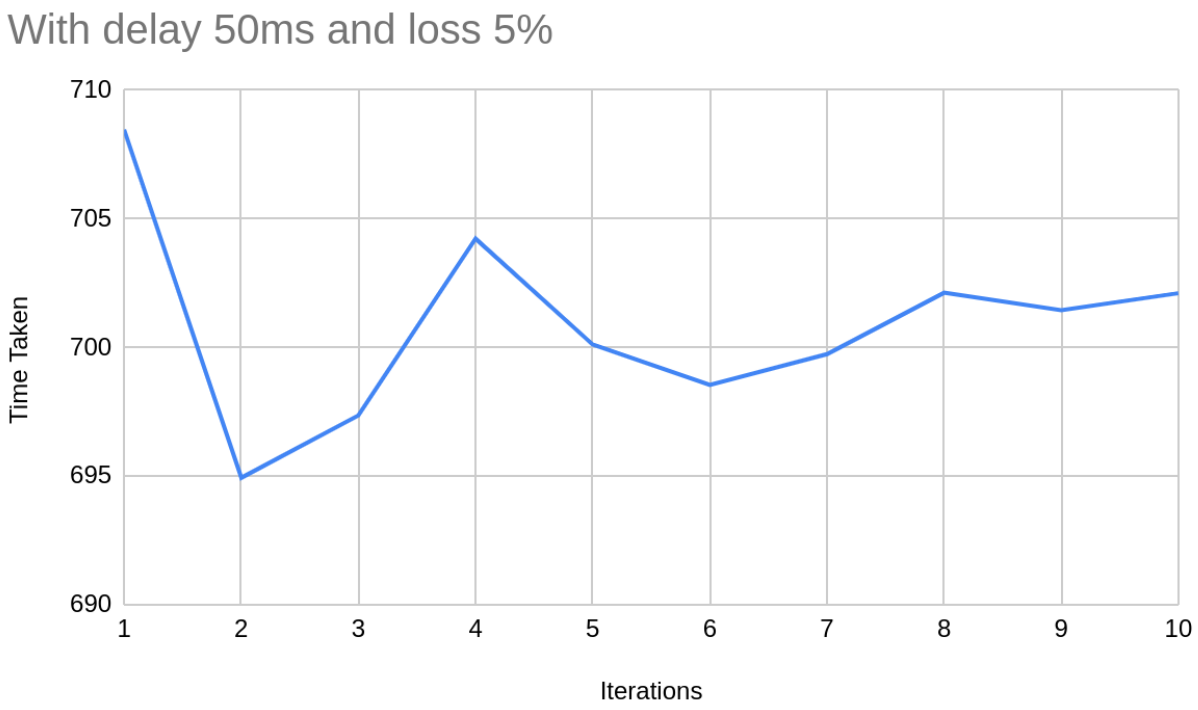
2) **Establish FTP connection in VM2 to VM1 for file transfer**

```
mahidharalam@server96:~$ ftp 192.168.122.182
Connected to 192.168.122.182.
220 (vsFTPd 3.0.3)
Name (192.168.122.182:mahidharalam): mahidharalam
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

3) **Recv the file from VM2 using recv filename command for 10 times**

```
ftp> recv CS3543_100MB
local: CS3543_100MB remote: CS3543_100MB
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for CS3543_100MB (104857600 bytes).
226 Transfer complete.
104857600 bytes received in 694.93 secs (147.3528 kB/s)
```

4) **Average Time Taken is 700.98s**



5) **Throughput**
   a) **Average effective Bandwidth/Throughput is 143KB/s ,Which is 1144Kbps**
   b) **File size is 104857600bytes==104857600*8bits**
   c) **Time =(FIlesize)/Bandwidth)**
      i) **104857600*8/(1144*1024)=716.08s**

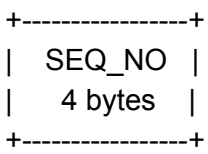
As the delay and packet loss can lead to retransmission and large RTT in TCPs the low throughput is observed in case2.

TASK 2

Packet Structure

Data Packet:

Neg - ACK Packet:



Features

1. Layered - any application can be built on top of this
2. Retransmission
3. Reliable
4. Acknowledgements

Protocol and Implementation

1. RDT is made as a modular protocol i.e any application can be built using it as a pseudo-transport layer protocol built on the UDP
2. RDT exposes the following APIs for the server and client side:
    a. Server:
        i. Bind – Bind the port and IP to the socket
        ii. Listen – Actively listen for the client connections
    b. Client:
        i. Connect – Used by client to connect to server

    Recv – receive the data of predetermined size
    Send – Send the data
3. Firstly in the connect step the address of the other end is saved and made the default destination address, so we can use the send and recv() functions instead of the sendto() and recvfrom()

4. In the connect step any information can be exchanged like the PKTSIZE, windowSize (implicit) etc..
5. In the current implementation the PKTSIZE is fixed but can be easily extended. The windowSize is used to determine the time to wait before sending the next batch of packets to avoid buffer overflow of the underlying UDP and miss packets. In the current implementation it is fixed based on the identified best value.
6. The recv and send functions expect the size of the message as an argument, hence both sides should follow some predetermined protocol to communicate the message length.
7. The send function works as follows, it computes the number of packets and other parameters to send. Next it sends the packets with data copied to temp buffer from the supplied data buffer. Here based on the window size a delay is applied between the packets.
8. After sending all the packets the send function waits for the negative acks, which contains the seq_no, so that it can resend the packet. If the seq_no contains -1 in the packet it implies that the message is received completely. Hence send function can return.
9. Receive function listens for the packets from the connected host, after receiving the packet it marks the corresponding the seq_no as received, after it stops receiving the packets from the other side,
10. It iterates over the unreceived seq_no, and send the n-acks for retransmission.

our-UDP-FTP- Implementation

1. The FTP client connects to the server (given IP and PORT)
2. FTP server binds the socket to chosen port and IP
3. FTP server maps the file to its virtual address space, using the *mmap* , and calls the rdt_send function specifying the address of the file mapped
4. FTP clients opens a file(creates file do not exist) and the truncates the file to its size and get it mapped to its virtual address space, which gets filled by the data in the rdt-layer.
5. Then it flushes the data to a file.

Without Delay and loss

1) RUN the command `sudo tc qdisc add dev enp1s0 root netem rate 100Mbit` in VMs

2) Recv the file to VM2

```
vm@vm:~/vm2/rdt2$ tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8003: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8824.590000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8801.455000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8820.075000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8827.203000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 9050.943000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8831.335000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8920.324000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8840.750000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8843.002000ms
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 8978.109000ms
vm@vm:~/vm2/rdt2$
```

3) Average Time Taken is 8.8734s

4) Throughput calculation :

- **d) Average Effective Bandwidth/Throughput taken is 11.29 MB/s which is 90.32Mbps**
- **e) File size is 104857600bytes==104857600*8bits**
- **f) Time =(FIlesize)/Bandwidth)**
  - **i) 104857600*8/(90.32*1024*1024)=8.85s**

Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.122.137	192.168.122.80	UDP	43	58780 → 8025 Len=1
7	0.081326974	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
13	0.081990893	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
19	0.082653477	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
25	0.083316053	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
31	0.083981636	192.168.122.80	192.168.122.137	WireGuard	738	Transport Data, receiver=0
37	0.084646815	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
43	0.085312107	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
49	0.085977059	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
55	0.086642649	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
61	0.087307601	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
67	0.087972844	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
73	0.088638012	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096

No.	Time	Source	Destination	Protocol	Length	Info
77695	9.037403490	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77701	9.038066089	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77707	9.038732234	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77713	9.039396560	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77719	9.040061808	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77725	9.040727319	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77731	9.041391731	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77737	9.042057989	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77743	9.042723005	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77749	9.043387730	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77755	9.044053142	192.168.122.80	192.168.122.137	UDP	738	8025 → 58780 Len=8096
77756	9.044120018	192.168.122.137	192.168.122.80	UDP	46	58780 → 8025 Len=4
77757	9.044136063	192.168.122.137	192.168.122.80	UDP	46	58780 → 8025 Len=4

As we can see from the packet capture using wireshark, the data transfer completed at 9.04sec, after that 4-byte acknowledgement packet to indicate transfer complete

PART 2:

1) RUN the command sudo tc qdisc change dev enp1s0 root netem delay 50ms loss 5% rate 100Mbit in VMs

2) Recv the file to VM2

```
vm@vm:~/vm2/rdt2$ tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8003: dev enp1s0 root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
vm@vm:~/vm2/rdt2$ gcc client.c rdt.c -pthread -lm -o client.out && ./client.out
Elapsed time: 522.931000ms
vm@vm:~/vm2/rdt2$
```

```
mahidharalam@server96:~/vm2$ gcc client.c rdt.c -o client.out -pthread -lm
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.641113ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 522.307800ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 522.307800ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 522.307800ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.863342ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.863342ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.974426ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.974426ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 522.307800ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 521.641113ms
mahidharalam@server96:~/vm2$ ./client.out
Elapsed time: 522.307800ms
```

3) Average Time Taken is 521.93s

4) **Throughput**
   - d) **Average effective Bandwidth/Throughput is 201.2KB/s ,Which is 1572.3Kbps**
   - e) **File size is 104857600bytes==104857600*8bits**
   - f) **Time =(FIlesize)/Bandwidth)**
     - i) **104857600*8/(1572.3*1024)=521.01s**