Computer Networks - II
prof. kotaro kataoka
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

Computer Networks 2

Ambati Sanjay Krishna - CS19BTECH11013

Uppala Sehouriey - CS19BTECH11015

Lingareddy Harshavardhan Reddy - CS19BTECH11023

Narayana Sreehitha Reddy - CS19BTECH11016

February 20, 2022

Contents

1 TASK-1 : Preparation and Preliminary Study

1.1 Implementing the test-bed with two directly connected VMs

Virt-manager : The virt-manager application is a desktop user interface for managing virtual machines through libvirt

- simply put virt-manager is a graphic tool for creating and managing guest virtual machines.
- We have created 2 VM instances in virt-manager by using ISO-image of an ubuntu-20.04 server.
- We will create a bridge on which these two VMs are placed ,

NETWORK DIAGRAM :

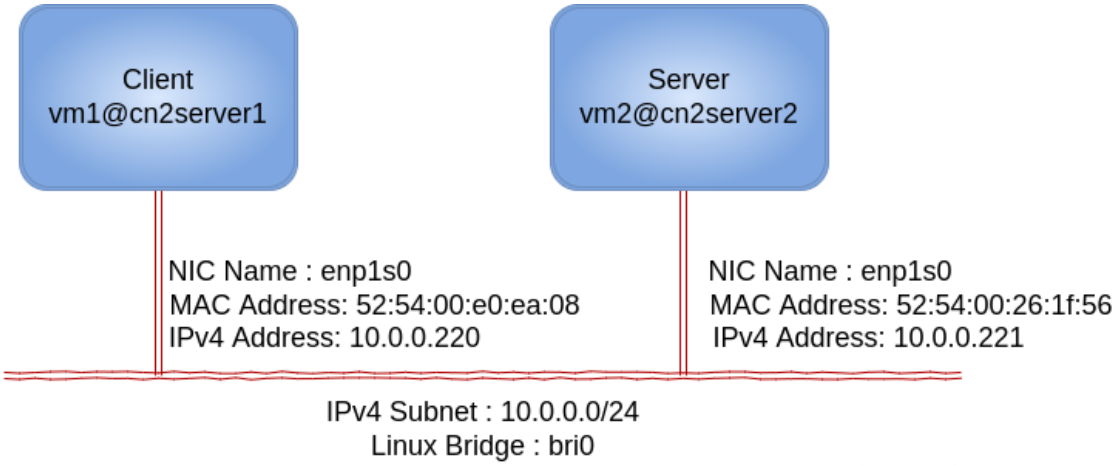


Figure 1: Network Diagram showing Linux bridge and two VM instances

Bridge Creation

```
# Creating a Linux bridge interface on Host Ubuntu
$ sudo brctl addbr bri0
# Making the interface up and running
$ sudo ip link set bri0 up
# Linux Bridge can be operated as a NIC
$ ip addr show bri0
# Removing the bridge created
$ Brctl delbr bri0
# To see the current bridges
$ brctl show
```

- Creating a Linux Bridge

```
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$ sudo brctl addbr bri0
[sudo] password for sehouriey:
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$ sudo ip link set bri0 up
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$ sudo brctl show
bridge name     bridge id               STP enabled     interfaces
bri0            8000.000000000000       no
virbr0          8000.525400f679f2       yes             virbr0-nic
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$
```

Figure 2: Creating the Linux Bridge using bridge control

- Adding both the VM instances onto the bridge

```
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$ sudo brctl show
bridge name     bridge id               STP enabled     interfaces
bri0            8000.fe5400261f56       no              vnet0
                                                        vnet1
virbr0          8000.525400f679f2       yes             virbr0-nic
sehouriey@sehouriey-ASUS-Gaming-FX570UD:~$
```

Figure 3: Two VM instances on the Linux bridge

- Showing Bridge connection in a VM instance in virt-manager

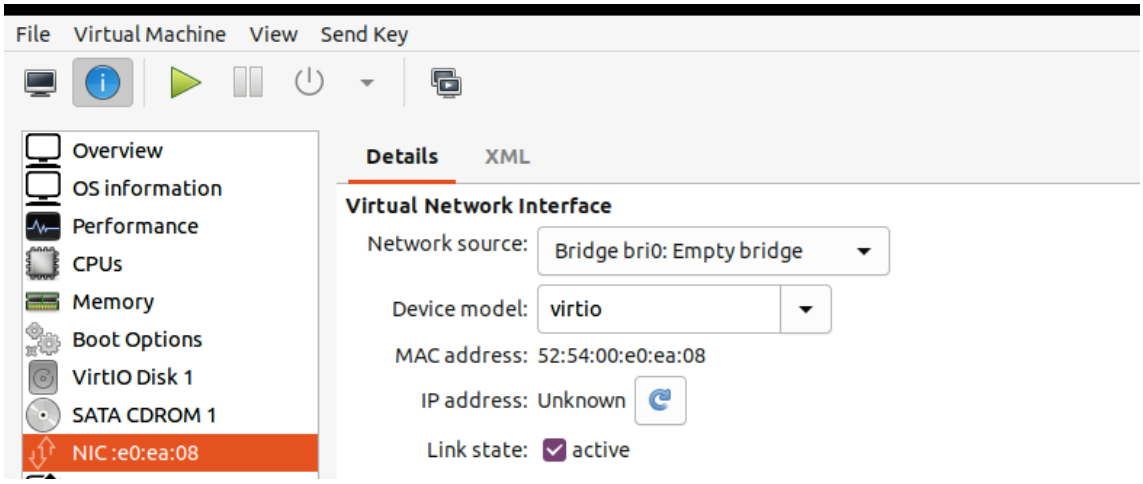


Figure 4: showing the setting of bridge connection in GUI of Virt-manager

1.2 Necessary Installations and tools

- Virt-manager
  Installation commands :

```
    $ sudo apt-get update
    $ sudo aptapt-get install qemu-kvm libvirt-clients libvirt-daemon-
   system bridge-utils virt-manager vlan

```

- tc
- vsftpd **very secure FTP daemon**
  - One of the most secure and fastest FTP server for UNIX-like systems
  - It is a default FTP server in ubuntu and is liscensed under GNU General Public License.

  Installation commands :

```
    $ sudo apt update
    $ sudo apt install vsftpd
    $ sudo service vsftpd status

```

- Follow the first 5-steps in the link for setting of FTP user

1.3 VM instances login

1.3.1 VM1 - Client

- **Client** : The client here represents the end-user. Client connects to the server and transfers files using vsftpd(is discussed here). The client ensures the total file is transferred based on the ACKs and/or NACKs received from the server.

```
cn2server1 login:
cn2server1 login: vm1
Password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-96-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 1.0


64 updates can be applied immediately.
23 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Sat Feb 12 14:01:02 UTC 2022 on tty1
vm1@cn2server1:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qle
    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 d
    link/ether 52:54:00:e0:ea:08 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.220/24 brd 10.0.0.255 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fee0:ea08/64 scope link
       valid_lft forever preferred_lft forever
vm1@cn2server1:~$ _
```

Figure 5: VM1 login

1.3.2 VM2 - Server

- **Server** : The server receives the files and ACknowledges the received files/packets. It waits till all the files/ file packets are received.

```
cn2server2 login:
cn2server2 login: vm2
Password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-96-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat 12 Feb 2022 04:17:44 PM UTC

  System load:  0.34              Processes:               127
  Usage of /:   53.1% of 6.82GB   Users logged in:         0
  Memory usage: 18%               IPv4 address for enp1s0: 10.0.0.221
  Swap usage:   0%


63 updates can be applied immediately.
23 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Sat Feb 12 14:01:07 UTC 2022 on tty1
vm2@cn2server2:~$ ip addr show
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
    link/ether 52:54:00:26:1f:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.221/24 brd 10.0.0.255 scope global enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe26:1f56/64 scope link
       valid_lft forever preferred_lft forever
vm2@cn2server2:~$ _
```

Figure 6: VM2 login

1.4 Using the tc command

- 'tc' is used to configure traffic control in the Linux kernel
- It allows us to add delay and packet loss, which we will be using in the further modules.

1.5 SubTask-1 (Performing 10 FTP attempts with 100Mbps link speed)

- Configuring the traffic control using tc with 100Mbps link.

```
    $ sudo tc qdisc ad dev enp1s0 root netem  rate 100Mbit
    $ sudo tc qdisc show

```

1.5.1 VM1 - Client tc set-up

```
vm1@cn2server1:~$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
[sudo] password for vm1:
vm1@cn2server1:~$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
vm1@cn2server1:~$ _
```

Figure 7: VM1 tc settings

1.5.2 VM2 -Server tc set-up

```
vm2@cn2server2:~$ sudo tc qdisc ad dev enp1s0 root netem rate 100Mbit
vm2@cn2server2:~$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
vm2@cn2server2:~$
```

Figure 8: VM2 tc settings

1.5.3 FTP login in VM1 Client

```
vm1@cn2server1:~$ ftp 10.0.0.221
Connected to 10.0.0.221.
220 (vsFTPd 3.0.3)
Name (10.0.0.221:vm1): sehouriey
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
```

Figure 9: Logging into FTP

1.5.4 File Transfer

- Lets see the file transfer in the first two attempts from client to server

```
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 8.48 secs (11.7961 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 (11.7276 MB/s)
ftp>
```

Figure 10: First two attempts of file transfer

- The next eight attempts of FTP file transfer.

```
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 (11.6927 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 8.47 secs (11.8086 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 (11.7258 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 8.48 secs (11.7992 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 8.47 secs (11.8042 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 8.51 secs (11.7516 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 8.54 secs (11.7066 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 8.55 secs (11.6892 MB/s)
ftp>
```

Figure 11: Showing file receiving on the server side

- Showing the file receiving on the server side in the first two attempts

```
vm2@cn2server2:/home/sehouriey/ftp/files$ ls -l
total 102404
-rw------- 1 sehouriey sehouriey 104857600 Feb 12 17:43 CS3543_100MB
vm2@cn2server2:/home/sehouriey/ftp/files$ ls -l
total 102404
-rw------- 1 sehouriey sehouriey 104857600 Feb 12 17:46 CS3543_100MB
vm2@cn2server2:/home/sehouriey/ftp/files$ _
```

Figure 12: Showing file receiving on the server side

1.5.5 Graphical and Tabular Representations

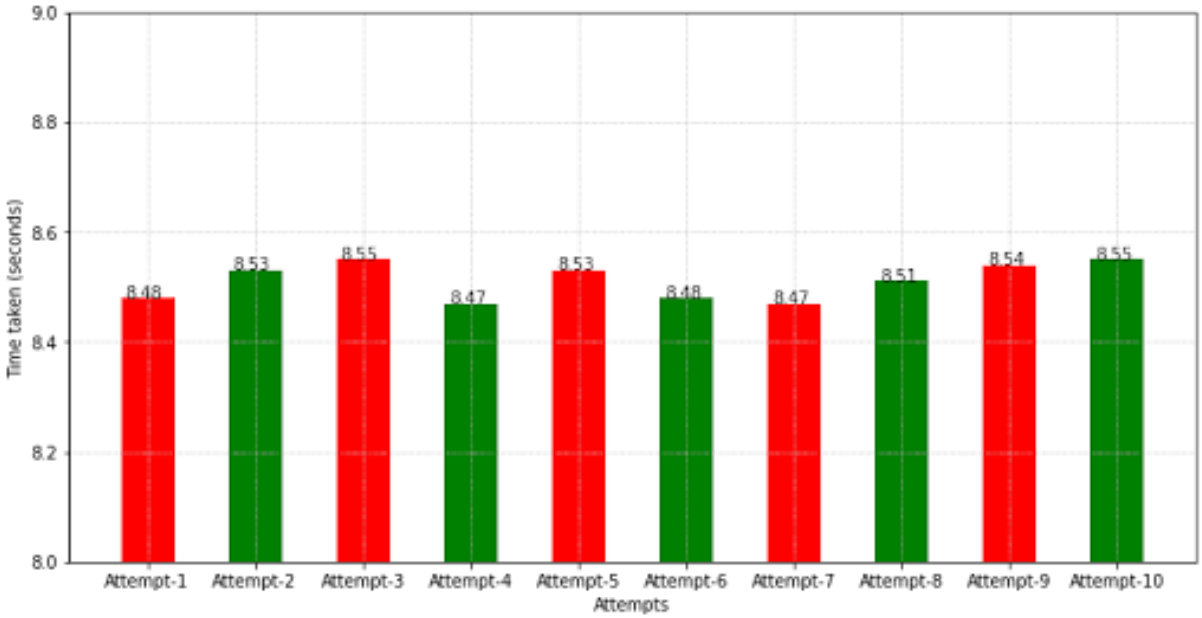


Figure 13: time taken with 100Mbps link speed and no delay or packet loss

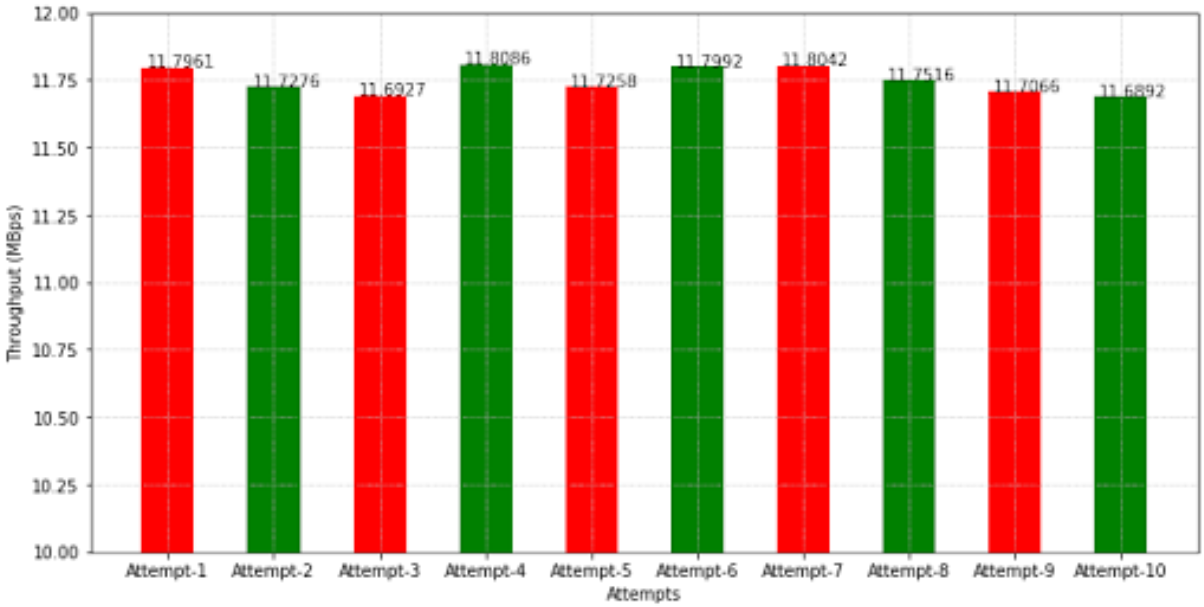


Figure 14: throughput with 100Mbps link speed and no delay or packet loss

Attempts	Time(in seconds)	Throughput(MBps)
Attempt-1	8.48	11.7961
Attempt-2	8.53	11.7276
Attempt-3	8.55	11.6927
Attempt-4	8.47	11.8086
Attempt-5	8.53	11.7258
Attempt-6	8.48	11.7992
Attempt-7	8.47	11.8042
Attempt-8	8.51	11.7516
Attempt-9	8.54	11.7066
Attempt-10	8.55	11.6892
Average	**8.511**	**11.75016**

1.5.6 Justification

- From the 10 attempts as observed in the above figures/tables:

- Average effective bandwidth = 117.5016/10
  = 11.75016 MB/s
  = 94 Mbps (100Mbps approx)

- Thus we can see effective bandwidth is close to the initial link rate 100Mbps that we have set using tc command.

- The average effective bandwidth is close to the initial setting as we have not added any delay or packet loss. This result changes in the next sub-task(section 1.5 justification) due to the different traffic configurations.
- Average time-taken = File size / average effective bandwidth
  = 100MB/(11.75016 MB/s)
  = 8.5105224099 s
- Time from tabular results = 8.511 seconds

Both the calculations have led to the same results reinforcing our interpretations and justification.

1.6 SubTask-2 (Performing 10 FTP attempts with 50ms delay and 5% packet loss)

- Configuring the traffic control using tc with 100Mbps link, 50ms delay, and 5

```
    $ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms
   loss 5%
    $ sudo tc qdisc show

```

1.6.1 VM1 - Client tc set-up

```
vm1@cn2server1:~$ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms loss 5%
vm1@cn2server1:~$ sudo tc qdisc show dev enp1s0
qdisc netem 8001: root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
vm1@cn2server1:~$
```

Figure 15: VM1 tc settings for 50ms delay and 5% packet loss

1.6.2 VM2 - Server tc set-up

```
vm2@cn2server2:~$ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms loss 5%
[sudo] password for vm2:
vm2@cn2server2:~$ sudo tc qdisc show dev enp1s0
qdisc netem 8001: root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
vm2@cn2server2:~$ _
```

Figure 16: VM2 tc settings for 50ms delay and 5% packet loss

1.6.3 FTP login in VM1 Client

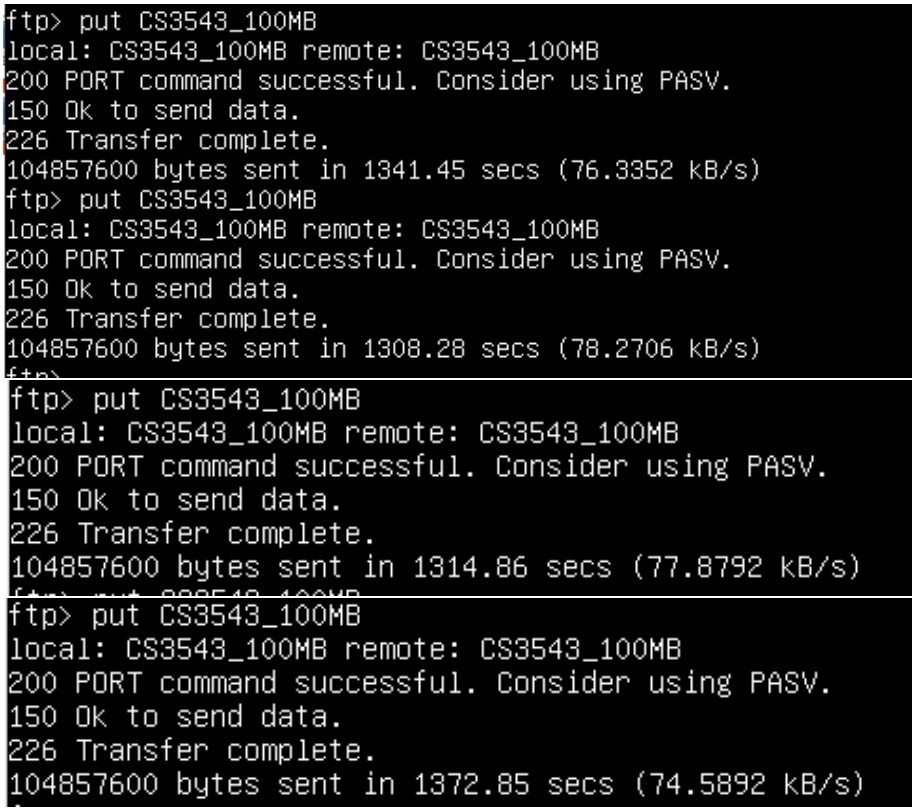
```
vm1@cn2server1:~$ ftp 10.0.0.221
Connected to 10.0.0.221.
220 (vsFTPd 3.0.3)
Name (10.0.0.221:vm1): sehouriey
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
```

Figure 17: Logging into FTP

1.6.4 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 1340.39 secs (76.3956 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 1378.16 secs (74.3022 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 1354.59 secs (75.5950 kB/s)
```



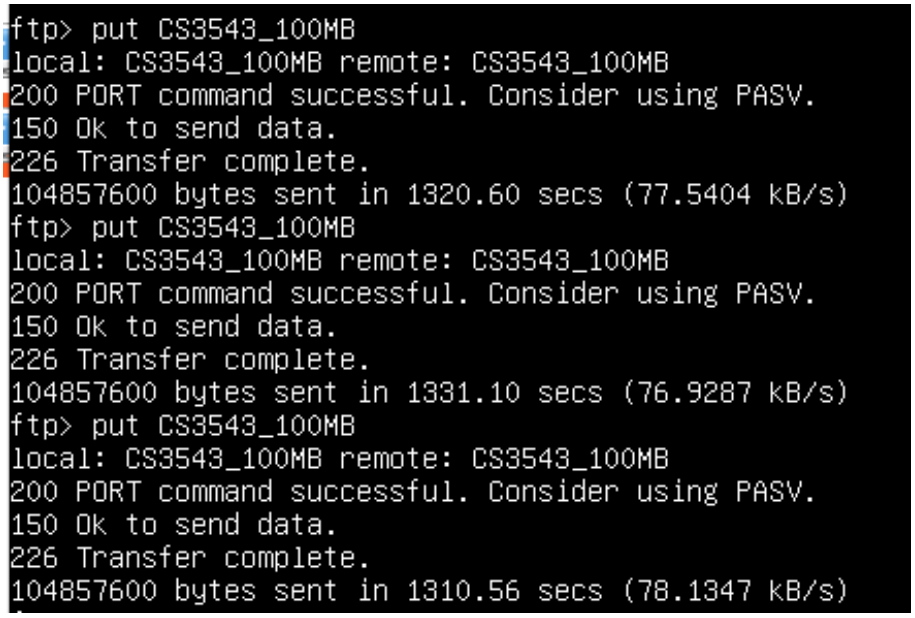


Figure 18: 10 attempts of FTP file transfer with 50ms delay and 5% packet loss

1.6.5 Graphical and Tabular Representations

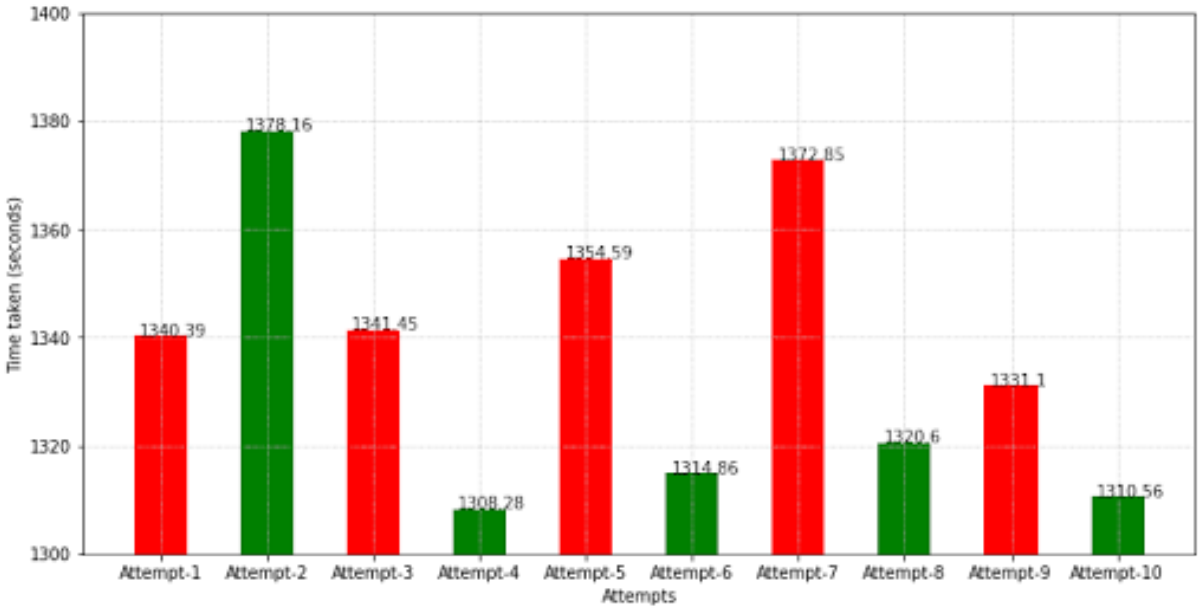


Figure 19: time taken with 100Mbps link speed and 50ms delay and 5% packet loss

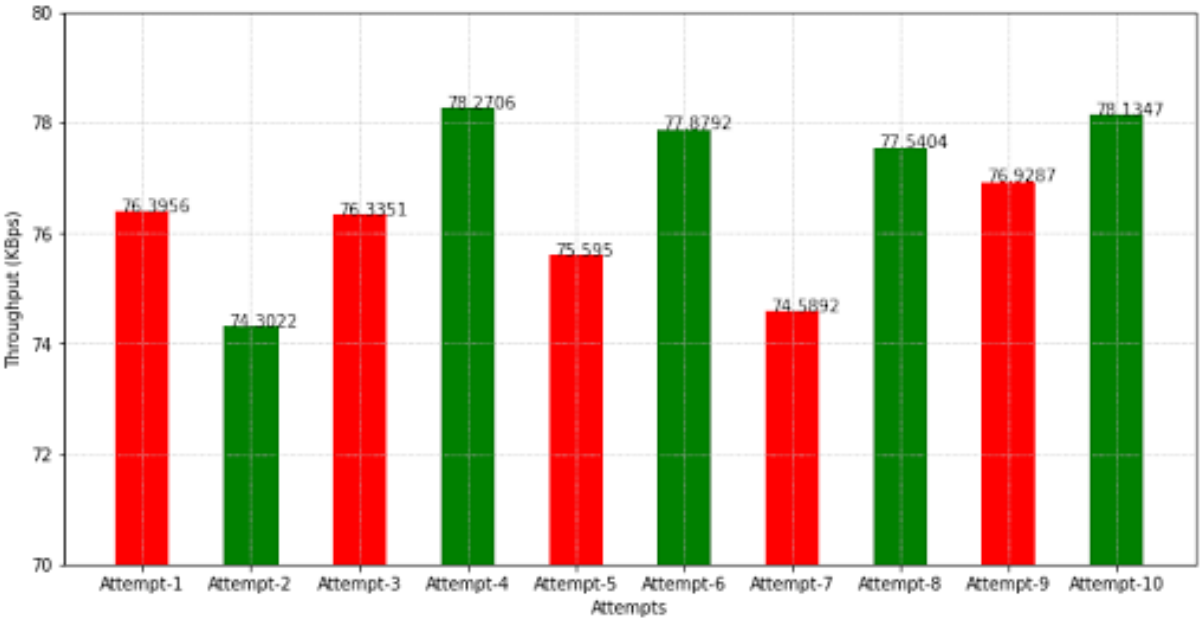


Figure 20: throughput with 100Mbps link speed and 50ms delay and 5% packet loss

Attempts	Time(in seconds)	Throughput(KBps)
Attempt-1	1340.39	76.3956
Attempt-2	1378.16	74.3022
Attempt-3	1341.45	76.3351
Attempt-4	1308.28	78.2706
Attempt-5	1354.59	75.5950
Attempt-6	1314.86	77.8792
Attempt-7	1372.85	74.5892
Attempt-8	1320.60	77.5404
Attempt-9	1331.10	76.9287
Attempt-10	1310.56	78.1347
Average	**1337.284**	**76.59708**

1.6.6 Justification

- From the 10 attempts as observed in the above figures/tables:
- Average effective bandwidth = 765.9708/10
  = 76.59708 KB/s
  = 0.075 MB/s
- We can see effective bandwidth is very far from the initially set link rate due to the packet delay and loss added.
- Average time-taken = File size / average effective bandwidth
  = 100MB/(0.075MB/s)
  = 1333.34 s
- Time from results = 13372.84/10 = 1337.284

Both the calculations have led to the same results reinforcing our interpretations and justification.

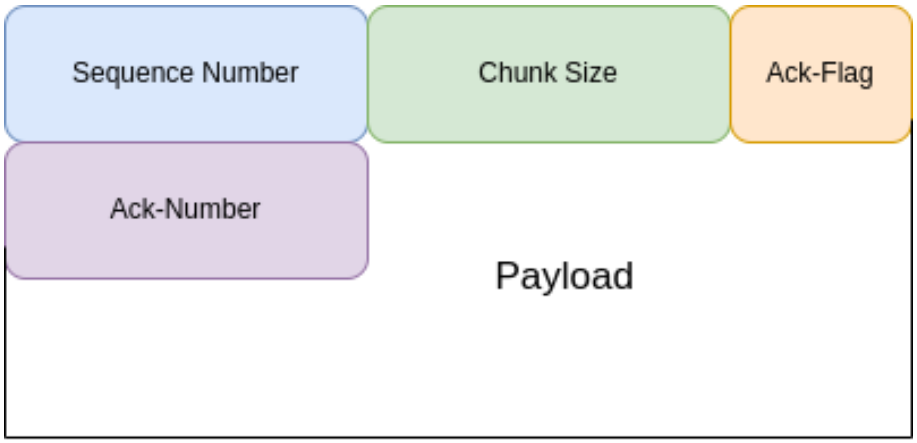
2 Task2: Implementing "Our-UDP-FTP"

"Our-UDP-FTP" is a simple UDP based FTP tool for application data transfer.

2.1 Packet Header and Pictorial Illustration:

The Packet Header consists:

- Sequence Number – 4-bytes(32 bits)
- Chunk size – 4-bytes(32 bits)
- Acknowledgement Number – 4-bytes(32 bits)
- Acknowledgement Flag – 2-bytes(16 bits)
- Payload – 8192-bytes



Total packet size is 9006-bytes.

2.2 Implementation of Algorithm in CPP:

- A Class is created on both sides i.e, for Client as well as for Server.
- A UDP Socket is created and is binded with given Port Number on the Server side.
- Whereas on the Client side, a UDP socket is created and is setted up with Server Address(i.e, Local Host Address) and given Port Number.
- On Providing correct parameters, the given file will be broken down into Packets
- Here 2 thread processes are used for sending and acknowledging the packets for fastness.
- On the Server side, the server will be waiting and will start receiving packets which are being sent from the Client side.

- After receiving a load packets, the server will send the acks to the Client.
- Basing on the acks received from the server, the Client will note the missing packets and re-send them using 2 threading processes.
- This looping continues until every packet reaches the Server.

2.3 Pictorial Illustration of Code working

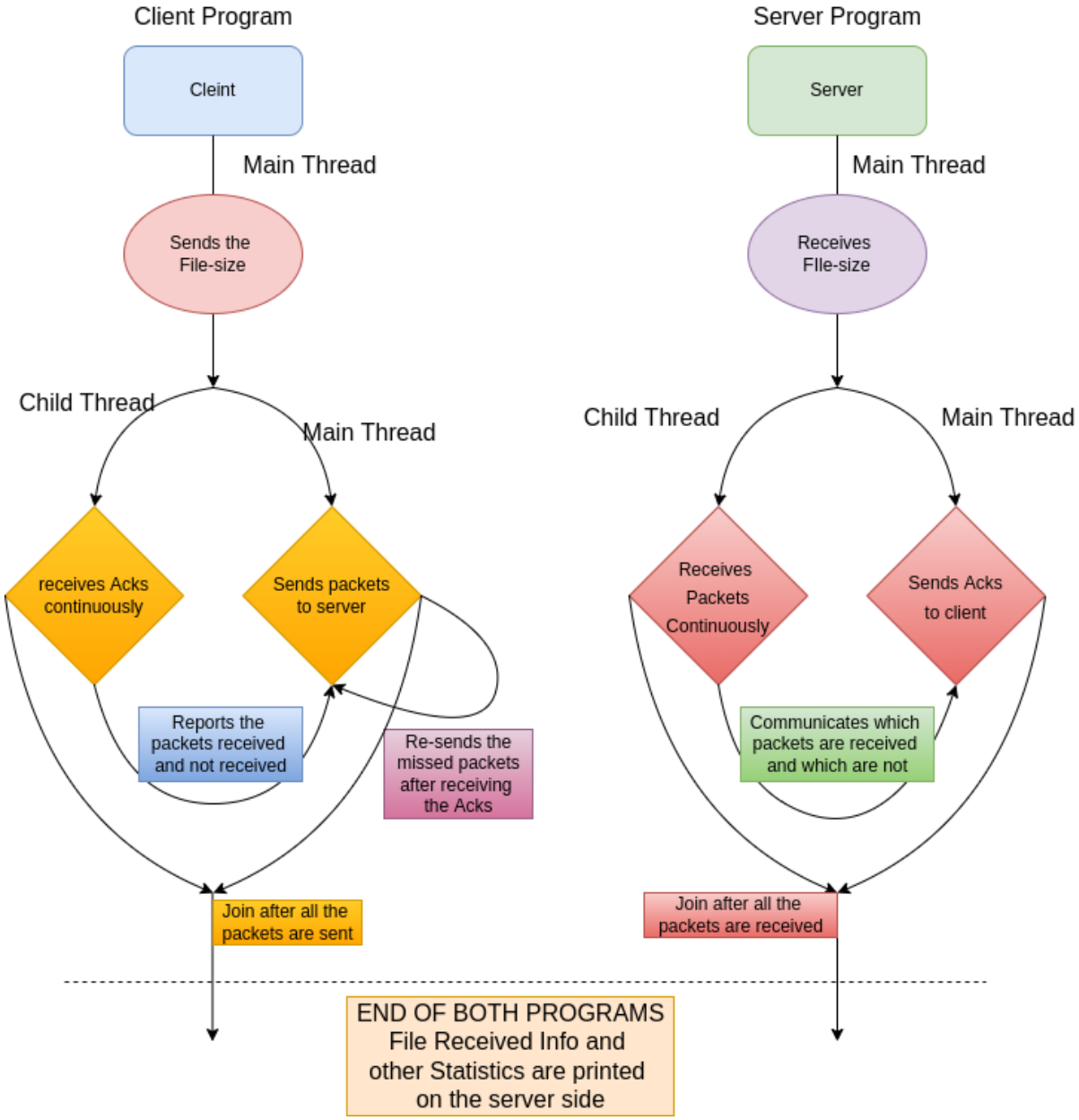


Figure 21: Program Algorithm

- Threads are maintained for high throughput.
- Acks for packets which are received on the server side are sent to client.
- Missing Packets are detected and re-transmitted until they are received.
- Controlled Flow of sending and receiving packets is maintained.
- Overall packet loss is detected and is set to minimal.
- Ordering of packets is done finally with help of sequence numbers.

2.4 RDT Features:

- Packet Loss Detection
- Acknowledgement
- Throughput
- Packet Re-transmission
- Flow Control
- Packets Ordering

2.5 SubTask-1 (Performing 10 attempts without delay and packet loss)

- Configuring the traffic control using tc with 100Mbps link.

```
1    $ sudo tc qdisc ad dev enp1s0 root netem  rate 100Mbit
2    $ sudo tc qdisc show
3
```

2.5.1 File Transfer and tc settings

- First attempt of File transfer, showing tc settings and file receiving.

```
vm2@cn2server2:~/cn2-seh$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
vm2@cn2server2:~/cn2-seh$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8007: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
vm2@cn2server2:~/cn2-seh$ ls -l
total 8
-rw-rw-r-- 1 vm2 vm2 4124 Feb 19 13:06 server.cpp
vm2@cn2server2:~/cn2-seh$ g++ server.cpp -o server -lpthread
vm2@cn2server2:~/cn2-seh$ ./server 5079
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.94219 seconds
Throughput: 11.1829 MBps
vm2@cn2server2:~/cn2-seh$ ls -l
total 102452
-rw-rw-r-- 1 vm2 vm2 104857600 Feb 19 13:29 received_file
-rwxrwxr-x 1 vm2 vm2     41208 Feb 19 13:28 server
-rw-rw-r-- 1 vm2 vm2      4124 Feb 19 13:06 server.cpp
vm2@cn2server2:~/cn2-seh$ _
```

Figure 22: server-side tc-settings and file receiving for first time

```
vm1@cn2server1:~/cn2-seh$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit
vm1@cn2server1:~/cn2-seh$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8007: dev enp1s0 root refcnt 2 limit 1000 rate 100Mbit
vm1@cn2server1:~/cn2-seh$ g++ client.cpp -o client -lpthread
vm1@cn2server1:~/cn2-seh$ ./client CS3543_100MB 10.0.0.221 5079
Size of the file sent: 104857600
Packet loss percent:0.132626
vm1@cn2server1:~/cn2-seh$ _
```

Figure 23: tc-settings and sending of file, client-side perspective1

```
vm2@cn2server2:~/cn2-seh$ g++ server.cpp -o server -lpthread
vm2@cn2server2:~/cn2-seh$ ./server 5080
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.98605 seconds
Throughput: 11.1284 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5081
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.91083 seconds
Throughput: 11.2223 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5082
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.93072 seconds
Throughput: 11.1973 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5083
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.92256 seconds
Throughput: 11.2075 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5084
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.93488 seconds
Throughput: 11.1921 MBps
vm2@cn2server2:~/cn2-seh$ _
```

```
vm2@cn2server2:~/cn2-seh$ ./server 5085
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.97116 seconds
Throughput: 11.1468 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5086
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.97063 seconds
Throughput: 11.1475 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5087
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.9365 seconds
Throughput: 11.1901 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5088
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 8.93233 seconds
Throughput: 11.1953 MBps
vm2@cn2server2:~/cn2-seh$
```

Figure 24: All 10 attempts of file-transfer with 100Mbit link speed

2.5.2 Graphical and Tabular Representations

Attempts	Time(in seconds)	Throughput(MBps)
Attempt-1	8.9421	11.1829
Attempt-2	8.98605	11.1284
Attempt-3	8.91083	11.2223
Attempt-4	8.93072	11.1973
Attempt-5	8.92256	11.2075
Attempt-6	8.93488	11.1921
Attempt-7	8.97116	11.1468
Attempt-8	8.97063	11.1475
Attempt-9	8.9365	11.1901
Attempt-10	8.93233	11.1953
Average	**8.943785**	**11.18102**

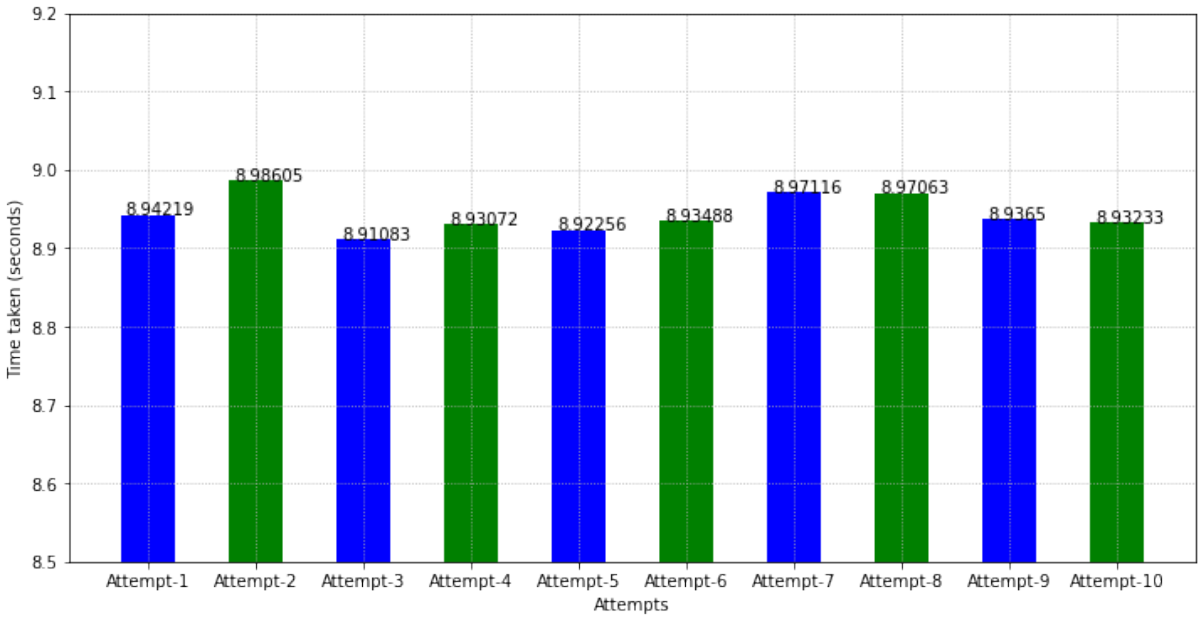


Figure 25: time taken with 100Mbps link speed and no delay or packet loss

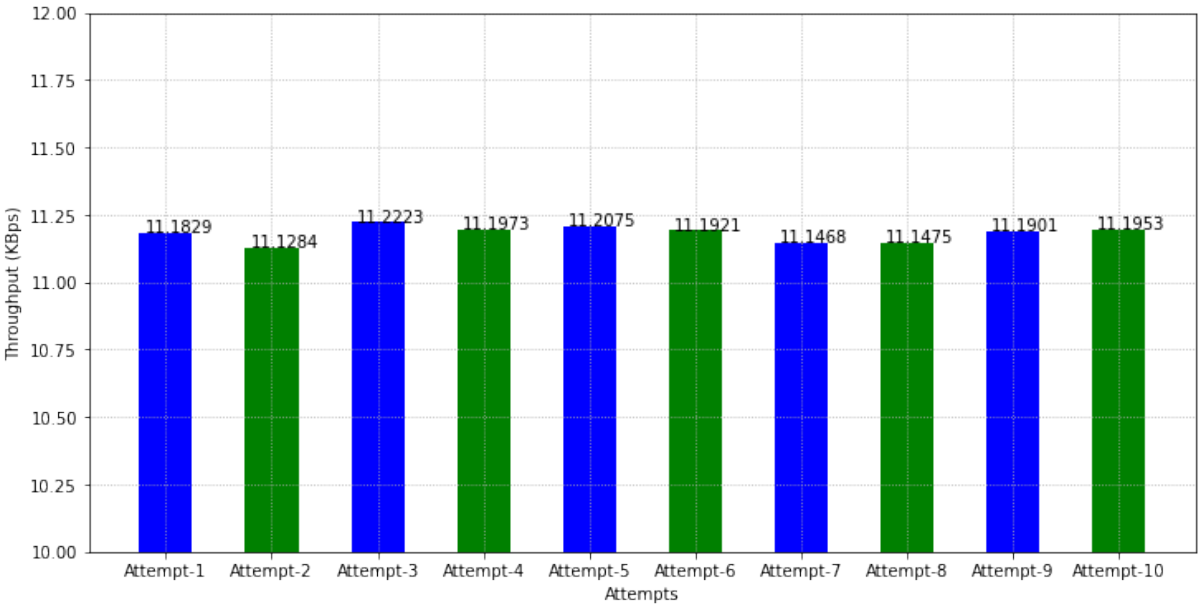


Figure 26: throughput taken with 100Mbps link speed and no delay or packet loss

2.5.3 Packet Capture with Wireshark

Time	Source	Destination	Protocol
5.452046101	10.0.0.220	10.0.0.221	UDP
5.453004296	10.0.0.220	10.0.0.221	UDP
5.453762124	10.0.0.220	10.0.0.221	UDP
5.454528764	10.0.0.220	10.0.0.221	UDP
5.455289604	10.0.0.220	10.0.0.221	UDP
5.456172040	10.0.0.220	10.0.0.221	UDP
5.456970709	10.0.0.220	10.0.0.221	UDP
5.457751467	10.0.0.220	10.0.0.221	UDP
5.458535009	10.0.0.220	10.0.0.221	UDP

Figure 27: packet capturing start

14.395285406	10.0.0.221	10.0.0.220	UDP
14.395427907	10.0.0.221	10.0.0.220	UDP
14.395590449	10.0.0.221	10.0.0.220	UDP
14.395665593	10.0.0.221	10.0.0.220	UDP
14.395761544	10.0.0.221	10.0.0.220	UDP
14.395804272	10.0.0.220	10.0.0.221	UDP
14.395837781	10.0.0.221	10.0.0.220	UDP

Figure 28: packet capturing end

Throughput calculation :
Throughput = FileSize/ time-taken
Throughput = (100MB)/ (14.3959 - 5.4520)seconds
Throughput = 11.1808 MBps (close to the average throughput we observed from experimenting)

2.5.4 Justification

- From the 10 attempts as observed in the above figures/tables:
- Average effective bandwidth = 11.18102 MBps
- Using ourUDPFTP packets gave almost same throughput with 100Mbit speed link as that of a normal FTP file transfer
- Average time-taken = File size / average effective bandwidth
  = 100MB/(11.18102)
  = 8.94372

- Time from results = 8.943785

Both the calculations have led to the same results reinforcing our interpretations and justification.

2.6 SubTask-2 (Performing 10 attempts with 50ms delay amd 5%packet loss)

- Configuring the traffic control using tc with 100Mbps link, 50ms delay, and 5

```
1    $ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 50ms
     loss 5%
2    $ sudo tc qdisc show
3
```

2.6.1 File Transfer and tc settings

- First attempt of File transfer, showing tc settings and file receiving.

```
vm2@cn2server2:~/cn2-seh$ sudo tc qdisc add dev enp1s0 root netem rate 100Mbit delay 50ms loss 5%
vm2@cn2server2:~/cn2-seh$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8002: dev enp1s0 root refcnt 2 limit 1000 delay 50.0ms loss 5% rate 100Mbit
vm2@cn2server2:~/cn2-seh$ ls -l
total 8
-rw-rw-r-- 1 vm2 vm2 4863 Feb 20 05:19 server.cpp
vm2@cn2server2:~/cn2-seh$ g++ server.cpp -o server -lpthread
vm2@cn2server2:~/cn2-seh$ ./server 5089
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 523.584 seconds
Throughput: 0.190991 MBps
vm2@cn2server2:~/cn2-seh$ ls -l
total 102452
-rw-rw-r-- 1 vm2 vm2 104857600 Feb 20 05:32 received_file
-rwxrwxr-x 1 vm2 vm2     41208 Feb 20 05:23 server
-rw-rw-r-- 1 vm2 vm2      4863 Feb 20 05:19 server.cpp
vm2@cn2server2:~/cn2-seh$ _
```

Figure 29: server-side tc-settings and file receiving for first time

```
vm2@cn2server2:~/cn2-seh$ g++ server.cpp -o server -lpthread
vm2@cn2server2:~/cn2-seh$ ./server 5090
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 523.001 seconds
Throughput: 0.191204 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5091
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 523.209 seconds
Throughput: 0.191128 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5092
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 524.214 seconds
Throughput: 0.190762 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5093
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 524.027 seconds
Throughput: 0.19083 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5094
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 524.607 seconds
Throughput: 0.190619 MBps
vm2@cn2server2:~/cn2-seh$ ./server 5095
Server socket created
Server Socket is succesfully binded
Whole File Received
Size of the file received: 104857600Bytes
Total Time Taken For File Transfer Using OurUDPFTP: 496.774 seconds
Throughput: 0.201299 MBps
vm2@cn2server2:~/cn2-seh$
```

Figure 30: tc-settings and sending of file, client-side perspective1

2.6.2 Graphical and Tabular Representations

Attempts	Time(in seconds)	Throughput(KBps)
Attempt-1	523.514	195.574784
Attempt-2	523.001	195.729896
Attempt-3	523.209	195.715072
Attempt-4	524.214	195.340288
Attempt-5	524.027	195.40992
Attempt-6	524.607	195.193856
Attempt-7	496.774	206.130176
Attempt-8	522.923	195.822592
Attempt-9	526.695	194.419712
Attempt-10	524.47	195.245056
Average	**521.3434**	**196.458**

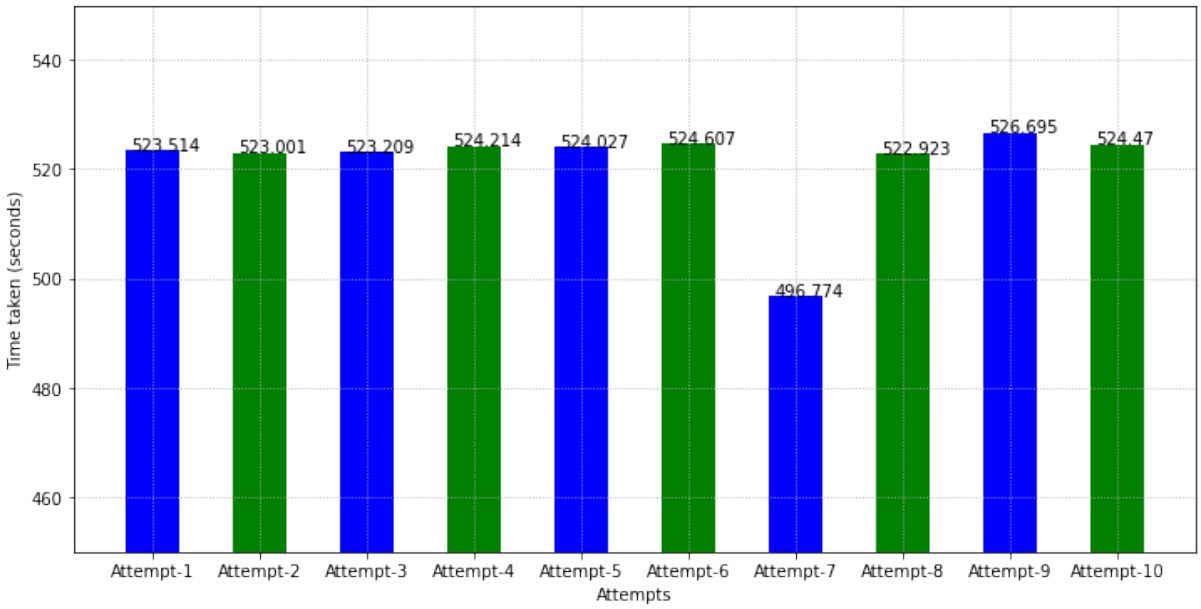


Figure 31: time taken with 100Mbps link speed with 50ms delay and 5% loss

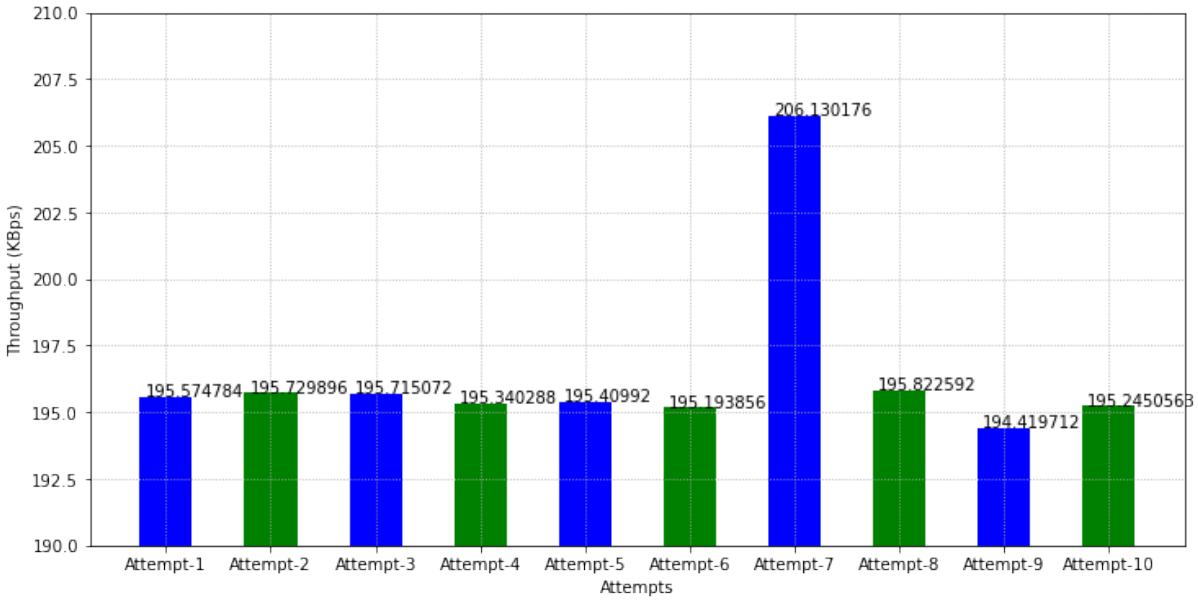


Figure 32: throughput taken with 100Mbps link speed with 50ms delay and 5% loss

2.6.3 Packet Capture with Wireshark

Time	Source	Destination	Protocol	Le
5.750244760	10.0.0.220	10.0.0.221	UDP	
5.752885023	10.0.0.220	10.0.0.221	UDP	
5.754356731	10.0.0.220	10.0.0.221	UDP	
5.755328791	10.0.0.220	10.0.0.221	UDP	
5.756067312	10.0.0.220	10.0.0.221	UDP	
5.756889308	10.0.0.220	10.0.0.221	UDP	
5.757747792	10.0.0.220	10.0.0.221	UDP	
5.759462565	10.0.0.220	10.0.0.221	UDP	
5.760384524	10.0.0.220	10.0.0.221	UDP	
5.761355979	10.0.0.220	10.0.0.221	UDP	

Figure 33: packet capturing start

```
527.093327275 10.0.0.220      10.0.0.221      UDP
527.093385298 10.0.0.221      10.0.0.220      UDP
527.093408447 10.0.0.221      10.0.0.220      UDP
527.093481322 10.0.0.221      10.0.0.220      UDP
527.093554650 10.0.0.221      10.0.0.220      UDP
527.093620582 10.0.0.221      10.0.0.220      UDP
527.093687282 10.0.0.221      10.0.0.220      UDP
```

Figure 34: packet capturing end

Throughput calculation :
Throughput = FileSize/ time-taken
Throughput = (100MB)/ (527.0937-5.75)seconds
Throughput = 0.1918 MBps
Throughput = 196.41553 KBps (close to the average throughput we observed from experimenting)

2.6.4 Justification

- From the 10 attempts as observed in the above figures/tables:
- Average effective bandwidth = 196.458
- We can see effective bandwidth is very far from the initially set link rate due to the packet delay and loss added.
- Average time-taken = File size / average effective bandwidth
  = 100MB/(196.458KBps)
  = 521.23 seconds
- Time from results = 521.3434 seconds

Both the calculations have led to the same results reinforcing our interpretations and justification.

3 Observations :

- vsftpd uses TCP connections to transmit the packets/files.
- ourUDPFTP uses UDP to transmit the packets/files.
- By Measuring the time taken, and throughput for different configurations, we have observed :

- When only 100Mbit link speed is used, general vsFTP performed better. But ourUDPFTP started performing better as we add delay and packet loss.
- TCP is slower but is a reliable protocol when compared to UDP. But in comparision UDP is much faster and efficient. Hence in a lossy environment UDP performs better. this can be seen from the above experiments we performed.
- The threeway handshake, flags and extra toll in TCP makes it more stable and reliable but is slow in a lossy environment.
- Making a 'UDP that is reliable' packet, proves to perform better in lossy environment. As we can see in SubTask-2(2.6), we can see that ourUDPFTP had better throughput and took less time to transfer the file compared to the FTP transfer in SubTask-2(1.6)

4 Working on more configurations

4.1 Performing 3 attempts with 20ms delay amd 2%packet loss

- Configuring the traffic control using tc with 100Mbps link, 20ms delay, and 2

```
1    $ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 20ms
     loss 2%
2    $ sudo tc qdisc show
3
```

4.1.1 File Transfer

```
vm2@cn2server2:/home/sehouriey/ftp/files$ cd
vm2@cn2server2:~$ sudo tc qdisc change dev enp1s0 root netem rate 100Mbit delay 20ms loss 2%
vm2@cn2server2:~$ sudo tc qdisc show
qdisc noqueue 0: dev lo root refcnt 2
qdisc netem 8001: dev enp1s0 root refcnt 2 limit 1000 delay 20.0ms loss 2% rate 100Mbit
vm2@cn2server2:~$ cd /home/sehouriey/ftp/files/
vm2@cn2server2:/home/sehouriey/ftp/files$ ls -l
total 0
vm2@cn2server2:/home/sehouriey/ftp/files$ ls -l
total 102400
-rw------- 1 sehouriey sehouriey 104857600 Feb 20 12:01 CS3543_100MB
vm2@cn2server2:/home/sehouriey/ftp/files$ _
```

Figure 35: receiving of files on server side

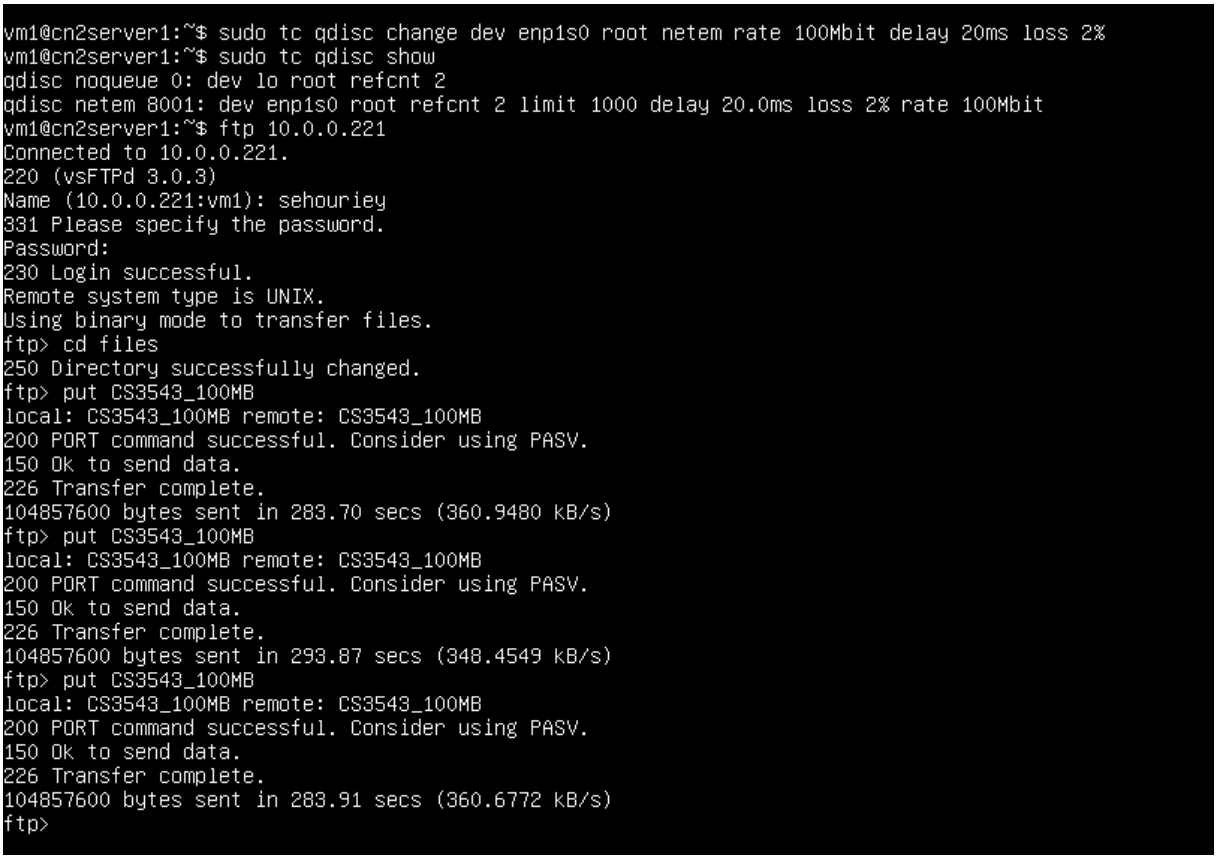


Figure 36: sending of files from client side

LaTeX generated document

**********END**********