EE 542 – Laboratory Assignment #4: Fast, Reliable File Transfer (Team)

Instructor: Young H. Cho

Due date: Sept 17 (Report) at 11:59pm and Sept 19 (Demo Video on Youtube)

In this lab, you will explore the real-world implications of how TCP performs under less than perfect conditions. Your goal is to develop a file transfer program (like scp or ftp) that uses your custom protocol that performs better than any other program available. This is your first competitive laboratory. Top 25% of the teams with highest performance will receive Extra credit. The program should be built for Linux OS and code your program using C or C++.

While TCP is effective, reliable, and relatively robust on the Internet, it doesn't always give us the best throughput under every circumstance. In this section you'll design an IP based file-transfer utility. The design and implementation of the utility is up to your group, however it must full-fill only three requirements: it must use IP (so it can be routed), it must transfer the file reliably (with no errors) and it must be implemented with a command-line interface similar to scp.

The link speed of the sender and receiver must be 100Mbps and the test file size must be at least **1GBytes**. You should emulate the delay and the loss rate of the link using the delay node. You should test your system under various different conditions. However, three settings that you must expose your system for the assignment are:

- The Delay (RTT) of 10ms with the Loss rate of 1% (bi-directional) set at router and link speed set to 100Mbits/sec for server, client and router. (case - 1)
- The Delay (RTT) of 200ms with the Loss rate of 20% (bi-directional) set at router and link speed set to 100Mbits/sec for server, client and router. (case - 2)
- Set link speed to 80 Mbits/sec and delay (RTT) to 200ms on with no drop at router and server, client link speed set to 100 Mbits/sec. (case - 3)

You should run the above tests with MTU of 1500 and then with MTU of 9000.

For your demo and competition, you will need to create an experiment with two nodes (VMs) connected by vyos router VM (which adds delay of RTT 200ms and the loss rate of 20% or 1%). So, in total 200ms should be visible for round-trip in ping and packet drop should happen in both Tx, Rx with Tx drop rate at 20% and Rx drop rate at 20% as per scenario given above.

Create a simple topology of server, client, Vyos VM in Virtualbox as done in lab1 (1 server VM <=> 1 Vyos VM <=> 1 client VM) and make your code working there first. **Do this to avoid the AWS cost since there are some restriction placed for AWS free tier if you use 3 VMs at same time. Once everything is working in VirtualBox, compile the same code in AWS ec2 instance and try out all the scenarios mentioned in the document and fix your code of any bugs encountered in AWS environment.**

For case-1,2: (record performance of your ftp code by transferring **1 Giga byte file**)

Set rate limit on both server and client VM to restrict speed to 100Mbit/s on egress by executing below command on both server, client. Here its assumed that eth0 belongs to 10.200.1.0/24 subnet for client and eth0 belongs to 10.200.2.0/24 subnet for server respectively.

Egress rate limiting:

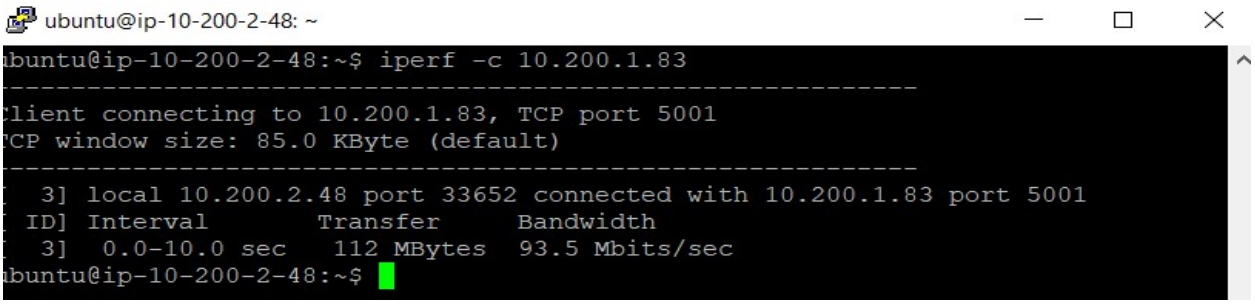
```
# sudo tc qdisc add dev eth0 root tbf rate 100mbit latency 0.001ms burst 9015
```

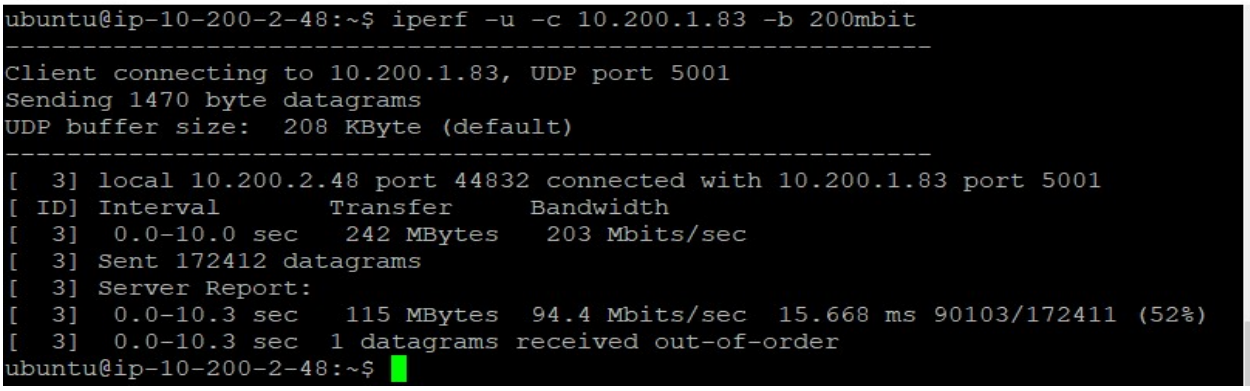
Now set rate limit to 100Mbit/sec on egress path of router on two network interfaces by executing below command. Here its assumed that eth0 belongs to 10.200.1.0/24 subnet and eth1 belongs to 10.200.2.0/24 subnet.

Egress rate limiting

```
# sudo tc qdisc add dev eth0 root handle 1:0 tbf rate 100mbit latency 0.001ms burst 9015
# sudo tc qdisc add dev eth1 root handle 1:0 tbf rate 100mbit latency 0.001ms burst 9015
```

First ensure that rate limiting is working by executing iperf. You need to get a tcp throughput of around 100 Mbits/sec. Also check the bandwidth on udp mode to be around 100Mbits/sec (iperf -u -c <dst_ip> -b 200mbit).





Now, set delay of RTT 10ms and loss rate of 1% in both tx, rx by executing below command on both eth0 and eth1 network interface of Vyos VM (router). Here eth0 has 10.200.1.0/24 and eth1 have 10.200.2.0/24 or vice versa. The following commands below simulate packet drop with high propagation delay on the link.

For RTT of 10ms, drop of 1%:

```
# sudo tc qdisc add dev eth0 parent 1:1 handle 10: netem delay 5ms drop 1%
# sudo tc qdisc add dev eth1 parent 1:1 handle 10: netem delay 5ms drop 1%
```

For RTT of 200ms, drop of 20%:

```
# sudo tc qdisc change dev eth0 parent 1:1 handle 10: netem delay 100ms drop 20%
# sudo tc qdisc change dev eth1 parent 1:1 handle 10: netem delay 100ms drop 20%
```

From ping command: (execute on server to ping to client and also from client to ping to server)
Ping -i 0.2 -c 200 <dest_ip>

Ensure that you are seeing the delay and drop rate which you have configured at router in the ping output with drop rate within range of +- 20% from the set value and delay value almost slightly greater than the set value. A snapshot is given below as reference. (This is for case-2)

```
64 bytes from 10.200.1.83: icmp_seq=194 ttl=63 time=201 ms
64 bytes from 10.200.1.83: icmp_seq=196 ttl=63 time=201 ms
64 bytes from 10.200.1.83: icmp_seq=197 ttl=63 time=200 ms
64 bytes from 10.200.1.83: icmp_seq=199 ttl=63 time=201 ms

--- 10.200.1.83 ping statistics ---
200 packets transmitted, 126 received, 37% packet loss, time 40450ms
rtt min/avg/max/mdev = 200.906/201.086/202.648/0.319 ms, pipe 2
ubuntu@ip-10-200-2-48:~$
```

You would also need to confirm the packet drop rate on udp mode on both direction with iperf. (start server iperf on client VM and run it and then start it on server VM and run it) (iperf -u -c <dst_ip> -b 100mbit) with +-20% variation allowed on drop rate. (This is for case-2). You should see an approximate drop of 20% with +-20% variation on the value.

Server => Client:

```
ubuntu@ip-10-200-2-48:~$ iperf -u -c 10.200.1.83 -b 100mbit
------------------------------------------------------------
Client connecting to 10.200.1.83, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.200.2.48 port 41704 connected with 10.200.1.83 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   120 MBytes   101 Mbits/sec
[  3] Sent 85471 datagrams
[  3] Server Report:
[  3]  0.0-10.3 sec  92.8 MBytes  75.9 Mbits/sec  15.723 ms 19307/85470 (23%)
[  3]  0.0-10.3 sec  1 datagrams received out-of-order
ubuntu@ip-10-200-2-48:~$
```

Client => server:

```
ubuntu@ip-10-200-1-83:~$ iperf -u -c 10.200.2.48 -b 100mbit
------------------------------------------------------------
Client connecting to 10.200.2.48, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.200.1.83 port 48437 connected with 10.200.2.48 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   120 MBytes   101 Mbits/sec
[  3] Sent 85470 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  92.6 MBytes  77.7 Mbits/sec   0.098 ms 19414/85469 (23%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
ubuntu@ip-10-200-1-83:~$
```

TCP iperf:

```
ubuntu@ip-10-200-2-48:~$ iperf -c 10.200.1.83
------------------------------------------------------------
Client connecting to 10.200.1.83, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.200.2.48 port 33690 connected with 10.200.1.83 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-23.9 sec   384 KBytes   132 Kbits/sec
ubuntu@ip-10-200-2-48:~$
```

You will also need to create a 1GB file in your sender computer. You can create this from the steps mentioned in Lab3 of scp part.

```
# dd if=/dev/urandom of=data.bin bs=1M count=1024
```

For case-3: (record performance of your ftp code by transferring **1 Giga byte file**)

Delete previous set configurations on qdisc of all VMs (verify it using sudo tc qdisc show). Now set rate limit to 100Mbits/sec on server, client VM. On Router VM set rate limit to 80Mbits/sec with RTT of 200ms. There is no drop configured here in this step. Ensure that ping show a delay of 200ms and rate is limited near to 80Mbits/sec for udp test in iperf. with no packet drop visible in ping.

Router VM command:
Egress rate-limiting:

```
# sudo tc qdisc add dev eth0 root handle 1:0 tbf rate 80mbit latency 0.001ms burst 9015
# sudo tc qdisc add dev eth1 root handle 1:0 tbf rate 80mbit latency 0.001ms burst 9015
```

Egress delay add in tx, rx:

```
# sudo tc qdisc add dev eth0 parent 1:1 handle 10: netem delay 100ms
# sudo tc qdisc add dev eth1 parent 1:1 handle 10: netem delay 100ms
```

Server, client VM egress rate limit command:

```
# sudo tc qdisc add dev eth0 root tbf rate 100mbit latency 0.001ms burst 9015
```

```
ubuntu@ip-10-200-2-48:~$ iperf -u -c 10.200.1.83 -b 100mbit
------------------------------------------------------------
Client connecting to 10.200.1.83, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.200.2.48 port 46053 connected with 10.200.1.83 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   120 MBytes   101 Mbits/sec
[  3] Sent 85471 datagrams
[  3] Server Report:
[  3]  0.0-10.1 sec  93.2 MBytes  77.8 Mbits/sec   0.111 ms 18984/85469 (22%)
[  3]  0.0-10.1 sec  1 datagrams received out-of-order
ubuntu@ip-10-200-2-48:~$
```

```
ubuntu@ip-10-200-2-48: ~
ubuntu@ip-10-200-2-48:~$ iperf -c 10.200.1.83
------------------------------------------------------------
Client connecting to 10.200.1.83, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.200.2.48 port 33688 connected with 10.200.1.83 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.9 sec  3.38 MBytes  2.60 Mbits/sec
ubuntu@ip-10-200-2-48:~$
```

Critical Thinking

As you can observe in Case 2 and Case 3, throughputs of iperf UDP are similar (~78Mbps) but TCP differ by up to 20x. Think about your configuration and describe in the report exactly what might be happening for Case 2 and Case 3 in terms of network buffering, packet loss, latencies, and window sizes.

Custom Fast and Reliable Protocol Design

Your program will need to reliably transfer the entire file from the sender to the receiver at the highest performance possible through this **bad network link for case-1,2,3**. Also ensure that your ftp application can work with different mtu of the interface. **Test it on mtu of 1500 and then on mtu of 9001**. Mtu needs to be set to same value for for all the network interfaces of VMs belonging to subnet 10.200.1.0/24, 10.200.2.0/24. (i.e in server, client, vyos VM). Your client ftp application should save the file in the location specified in the command line. Similarly, server part of ftp application should send the file specified in the command line.

You must timestamp right when the first bit of the data leaves the sender and the final bit is received by the receiver. The total time for the file to make the one-way trip will be used to compare your program's performance against other team's program performance. You also will need to prove the reliability of your system by **running MD5 on both the original file and the received file**. You will need to run your ftp application against mtu of 1500 and 9001 on two specified drop rate, delay and rate limit as mentioned above for three cases. (repeat same procedure one with mtu of 1500 and then with 9001 and observe file transfer time)

The execution of the system should be straight forward, and any notion of cheating will result in automatic defeat in the competition and possibly zero on the lab score. The goal of this task is to encourage a healthy competitive development environment for everyone. You are encouraged to help each other to get the best result. However, at the end, two teams with highest throughput will receive extra credit.

1) The minimum transfer rate for your final FTP is 20 Mbps to compete against the other team.
2) Describe, in detail, the concept(s) behind your file transfer utility, results, flow chart, data structure and algorithm used and the analysis in the report document that must be submitted along with source files on the due date specified on blackboard by 11:59pm.
3) The scoring will be based on the submitted report slides and demo

REFERENCE

https://computing.llnl.gov/tutorials/pthreads/ (multi-threading)
https://www.geeksforgeeks.org/data-structures/ (data- structure)
https://www.geeksforgeeks.org/fundamentals-of-algorithms/ (algorithms)