CENG435 Group 25 TP Part-1 Report

Ali Alper Yüksel, Student ID: 2036390
Yunus Emre Gürses, Student ID: 2099083

1. Processes of Part-1

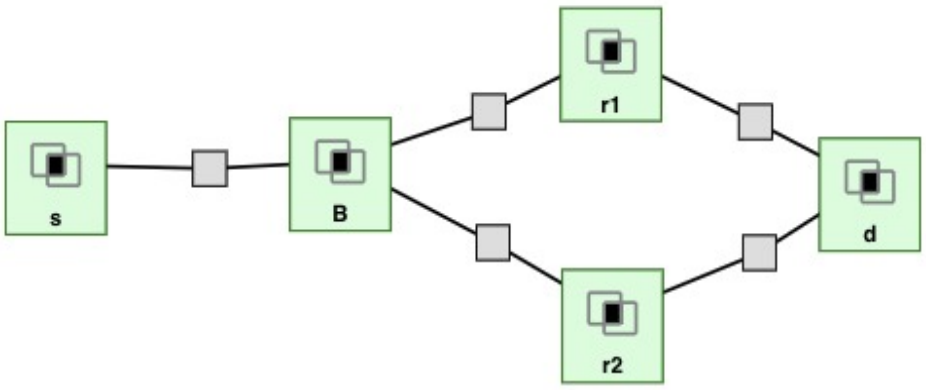
In this part-1 of project, we followed these steps:

1. Learning how to use Geni platform and increase our usage skills of Geni
2. Understanding the topology and implementing our network system depending on this topology
3. Using Python socket programming to build system
4. Uploading codes into virtual machines
5. Testing communication
6. Using netem commands to test delays and their relationships

2 .Geni Platform

After taking our Geni account, with lab-0 and lab-1 homeworks of Geni platform, we get experienced about how to use Geni platform. With that experience, we learnt how to build a topology and how to manage virtual machine in this topology via SSH.

3.Topology



As can be seen in topology, we have five hosts. s stands for source node, d stands for destination node, r1 and r2 stands for router nodes and B is considered as broker. So, explanations of components of this topology are following:

1. s (Source Node): Acts as source device which collects data as sensor measurements. It just sends data to broker (B) with TCP.
2. B : Acts as broker. It listens s and receive data when it is sent from source with TCP and packetize it. Then, send these packets to r1 and r2 with UDP.
3. r1 and r2 (Router Nodes): They listen b and take packets when it is sent from broker and send these packets to d with UDP.
4. d (Destination Node): It listens r1 and r2 and when packets are sent, d receives these packets.

The tables below show routing tables for r1 and r2. We can understand where is the next station to send for reaching our packet to one.

Destination	**Send To**
s	B
B	r1
r1	d

Table 1: $r1$ Route Table

Destination	**Send To**
s	B
B	r2
r2	d

Table 2: $r2$ Route Table

4. Programming Part

Since, there is no restriction about choice of programming language which we use, we used Python for this assignment in programming part. Python has more useful modules and its syntax needs less effort than other programming languages.

We used one Python library called as socket for socket programming. To calculate and work on time data, one Python time module is used which called as datetime, and one Google library is used which called as ntplib. ntplib also supply synchronization of time data.

1. socket: When sending message and creating socket, this module is used. When we create socket object, second parameter takes `SOCK_STREAM` constant to represent TCP, and to represent UDP, it takes `SOCK_DGRAM` constant.
2. ntplib: We use this library to take time data from Google servers as external.
3. datetime: We use this module to convert and represent types of time data which is received from Google servers.

5. Uploading codes into virtual machines

After writing codes into five seperate files, we uploaded files into virtual machines by using SSH. Also we used VIM editor to edit these python script files.

6. Testing communication and delays

After running programmes from s to d, we tested whether data is packetized and sent to destination node or not from source node firstly.

To calculate delay we encrypted current time in source node over data and after destination node receives packet successfully, we took current time in destination node as well. Their subtraction gives our end-end delay.

By doing this step, we had difficulty about synchronizing the time. For solving this problem, we used ntplib to synchronize all machines into a base time.

Also to be able to get more sensitive end-to-end delay value we created 50 data in source node and we calculated end-to-end delay. This means we executed the code more than one time. In the end, we took mean of these values and get more spesific end-to-end delay value.

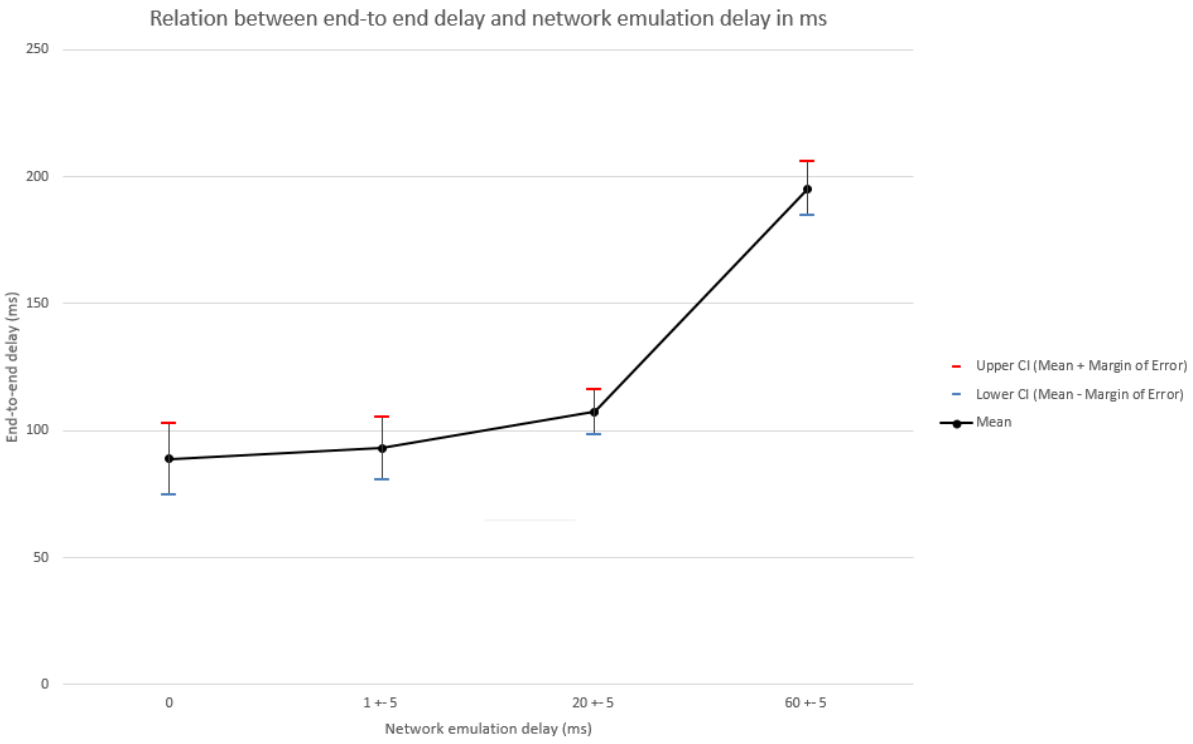
7. Using Netem Commands

We used netem commands to simulate and build our graph about relation between end to end delay and network emulation delay .

To obtain normal distribution 1ms $\pm$ 5ms, 20ms $\pm$ 5ms and 60ms $\pm$ 5ms, we added following command into virtual machines:

```
tc qdisc change dev eth0 root netem delay 60ms 5ms distribution normal
```

For broker node, we applied eth1 device to use link between broker and r2 and applied eth2 device to use link between broker and r1.
For r1 and r2 nodes, we applied eth2 device to use link between destination and r1 and r2 respectively.
For destination node, we applied eth1 device to use link between destination and r2 and applied eth2 device to use link between destination and r1.



In virtual machines, we calculated end-to-end delay over links, then we added the emulated delays which are 1ms $\pm$ 5ms, 20ms $\pm$ 5ms and 60ms $\pm$ 5ms with tc command. As in the shown in the graph, while emulated delays are added, end-end delay shows almost same behavior with small differences. This small difference was because of that propagation delay was not changed while processing delay is increased.