Batyr Charyyev 2001527 Ceng-435 HW1 Report

1. Introduction

Objections of our homework was to write basic packet based UDP socket application and conduct experiment with help of 'tc' command to calculate the end-to-end delay. Also we were familiarized with design of packet.

2. Design

We chose C language as a programming language. Our design was like node A(client) sends to node B(router) and it sends to C, C sends to D, D sends to E(server) via eth0(we consulted with assistant whether it will be problem to use eth0 and we get "it will not be problem as long as your program is stable") as an answer. We did not get any problems while we conducted experiments and we got reasonable results, so we continued with eth0. We get our data and IPs of all nodes through input.txt file. Our packet consist of header part and message part which ends with * character.
In header we have index of source IP, index of destination IP and all IPs separeted by delimiter. Each packet that we sent was 70 bytes which includes header and message. When we sent last message server knows that it is last message from * character of our message and sends acknowledgement message back to client. From routing tables you can see how our nodes sent packets to other nodes.

Routing Tables

Table 1: Routing Table A and Routing Table B

Destination	**Send To**	**Destination**	**Send To**
A	-	A	A
B	B	B	-
C	B	C	C
D	B	D	C
E	B	E	C

Table 2: Routing Table C and Routing Table D

Destination	**Send To**	**Destination**	**Send To**
A	B	A	C
B	B	B	C
C	-	C	C
D	D	D	-
E	D	E	E

Table 3: Routing Table for Node E

Destination	Send To
A	D
B	D
C	D
D	D
E	-

In routing table of Node C if our packet has destination of A,B then it is send to B, if destination is D,E then it is send to D. In our code we determine our destination with help of sourceIndex and destinationIndex in header part of message. The implementation of it, is well clarified in ReadMe file.

3. Implementation

How to run the code and how the code works is provided in ReadMe file so I would like to skip this steps.

4. Experiment

While we conducted experiments we used "tc"(traffic control) command to change emalutaion delay. Also we used "ntp"(network time protocol) to synchronize all nodes time.

tc commands:

```
sudo tc qdisc add dev eth0 root netem delay 10ms        to add delay
sudo tc qdisc change dev eth0 root netem delay 10ms     to change delay
sudo tc qdisc delete dev eth0 root netem                to delete delay
```

ntp commands:

```
sudo service ntp stop                        to stop ntp
sudo ntpdate -u 0.north-america.pool.ntp.org to synchronize
sudo service ntp start                       to start ntp
```

results of our experiment is provided below

10 +- 5ms emaluation delay 42.8937ms end-to-end delay
20 +- 5ms emaluation delay 81.8760ms end-to-end delay
40 +- 5ms emaluation delay 166.58553ms end-to-end delay
80 +- 5ms emaluation delay 332.2583ms end-to-end delay

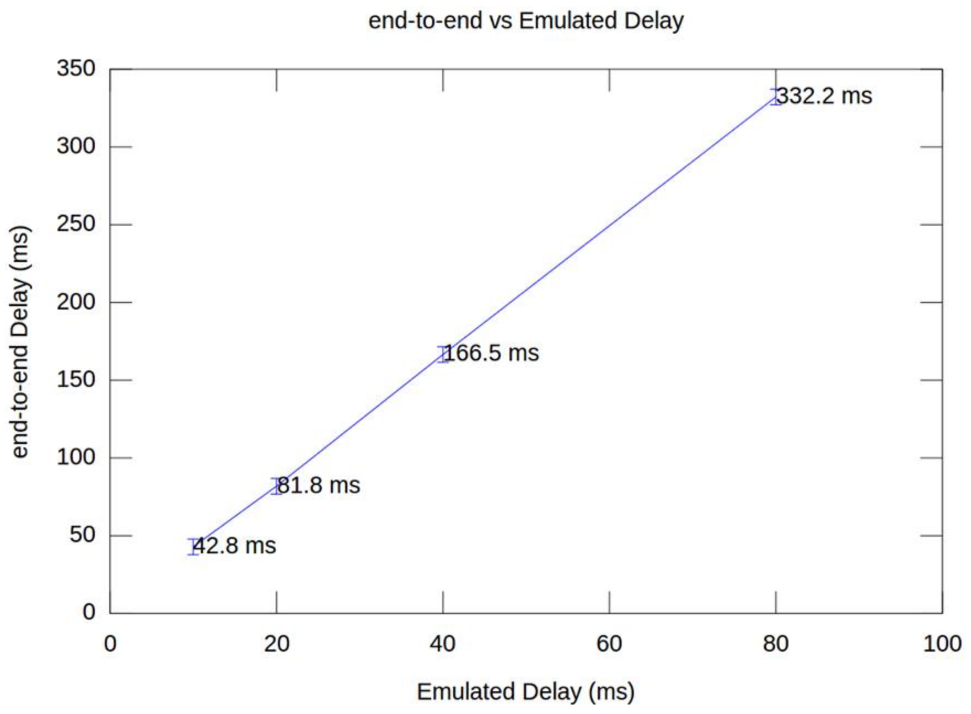

Figure 1: A simple caption

This graph shows the emaluation delay versus end-to-end-delay relation. We did the each experiment 10 times and took mean value of results. We can see that for 10ms emalutation delay we got 42.89ms end-to-end delay. From defintion of end-to-end delay which is: $numberOfLinks * (propagationDelay + processingDelay + transmissionDelay)$ We can say that since we have four links it is reasonable for end-to-end delay to be 4 times larger than emalutaion delay.

5. Conclusion

From definition of end-to-end delay, delay results because of transmission and propogation and proccessing parts also congestion in network results in different delays that is why we conducted experiment 10 times and took mean of it. The mean value is also as we expected.
What can be changed? We analyse and change the header part of our message in every node which corresponds to processing delay so we can make it as minimum as possible to minimize our processing delay and end-to-end delay.