CENG 435 Term Project Part-1 Group-26 Report

Sinan Talha KOSAR
Computer Engineering
Middle East Technical University
ANKARA
e2099190@ceng.metu.edu.tr

FURKAN DOGAN
Computer Engineering
Middle East Technical University
ANKARA
e2098937@ceng.metu.edu.tr

***Abstract*— In the first part of the term project we are asked to send our messages over unreliable network with TCP and UDP based socket application to the routers then catch the message from destination node. Briefly, send our message over TCP stream and convert it to UDP Datagram and vice versa. In this part, we have tested our scripts on Virtual Machines via GENI Platform. We have used NTP to synchronize the time between VMs. After that we have examined the end-to-end delay for the packets that we sent from source to destination.**

***Index Terms*—TCP, UDP, NTP, Broker, Router, Interface, IP, Port, end-to-end delay, netem-tc**

I. Introduction

In the first part of the term project we are given a network, an unreliable one, which consists 5 hosts:
a source node
a broker node
2 router nodes
and a destination node

Between these nodes we are given 5 nodes with configurations(IPs etc.). And we are asked to implement TCP connection between source and broker node, UDP connection between broker, router and destination nodes. To implement this configurations to given nodes we have used Python's socket module and embedded the scripts into nodes which works as given specifications. After configuring network and importing into GENI platform, we have send 100 messages to make our experiment more real-like and we have recorded the end-to-end delay (see section XX to see the calculation of end-to-end delay). Then we have examined the outputs. Detailed explanation of how codes work are described in ReadMe section and also in comments in that, please check it to understand rest of the report.

II. Design Part

We have already given a topology of a network that we will build our experiment on. Visual examination of topology is shown below (Fig. 1.)

*Note : All of the examples below are from code that we have written for the source node and one of the router

Each node has interfaces for incoming and outgoing connections with unique IP addresses. We are asked to design TCP connection between source and broker and UDP connection at remaining. We have used python socket module

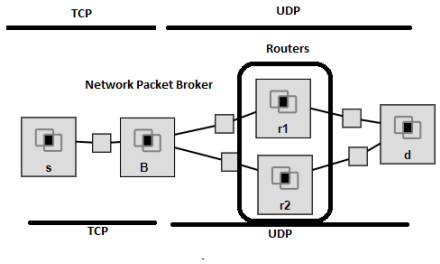

Fig. 1. The structure of topology

in-order to do the job. Example usages for both TCP and UDP in python socket module is shown below:
.socket(socket.AF_INET, #Internet (IPv4)
socket.SOCK_STREAM) => For TCP
.socket(socket.AF_INET, #Internet (IPv4)
socket.SOCK_DGRAM) => For UDP
We are using socket.AF_INET (Since we have given specification "IPv4"). So it expects a 2-tuple: (host, port).

After making necessary coding, we have managed to sent messages between source and destination. Since the main goal of this experiment is working on end-to-end delay and examine them, we have decided to send time as message. By doing so, we are able to sent message and calculate the end-to-end delay by using our message.

We have faced a problem at this stage, the clocks for both source and destination are not synchronized.
To over come, we have used Network Time Protocol (NTP), which is a protocol used to synchronize computer clock times in a network. Online, there is lots of NTP servers available, so to decide which to use, we have used "ping" command for each NTP server. "ping _IPaddress of NTP Server_ sends a package to server and waits for response and we are able to see the round-trip time in milliseconds for each ping. So Google's NTP server seems more faster to us than the others. We have decided to use NTP server of Google (time.google.com).

To use in python script, the is a library called "ntplib", after importing it usage is below:
c = ntplib.NTPClient()
response = c.request('time.google.com')
timer = response.tx_time

$*$ While using NTP Server to synchronize the clocks, we have taught whether the delays between source-NTP and destination-NTP are same? Just to be sure, we used ping command on both source and destination VMs, and difference between round-trip time is nearly at most 0.1 ms, which we will neglect from now on.

In our network packets must be send as strings. So, we have convert time to string then send it.
$>>$ MESSAGE = repr(timer)

We have sent 100 messages from source to broker and from broker we have sent same 100 messages to router 1 and router 2. At the end, 200 messages came to our destination.

$*$ Sending 100 messages are decided by ourselves, since it is an experiment, the more message we sent, more accurate examination we can done.

At this point, we have faced another problem. When we try to request time for each packet which is 100 times. NTP server times-out because of lots of requests. To overcome, we have added time.sleep(1) between each 10 messages.

Following paragraph is operative for all node's script although it's focusing on broker node.

There are ports for each sender nodes and listeners. The port for sender and receiver, which are operating between, must be same to catch the packet. In general, random free ports are from 1024 to 65535. That's why we have picked port numbers between these interval. Our message which consists time as string should be converted to float to be able to make calculations on them.
Briefly, our scripts work like:
Source node establish TCP connection to broker, takes time from NTP Server, converts it as string and send it.
Broker accepts TCP connection, gets the packets and establish UDP connections to router 1 and router 2 then forwards the packets.
Each router accept UDP connection, and forwards packets to destination node by establishing UDP connection.
Finally, destination node accepts UDP connections and gets the packages. Immediately after, takes the time from NTP server and compare the time to find end-to-end delay.
Design of Broker
In the broker, we are accepting packets with TCP and forwarding packets with UDP. To handle both of these in the same script we had to create two different sockets. To accept, we need to listen source node in same port that source node sends packets which is 12001 in our code and we need the IPs of interfaces belong to router 1 and router 2 and their port. We have accepted packet from source node and take it's message, then forwards by using socket which is implemented for UDP, to forward the routers. To forward, again we need to forward packets 10 by 10 to overcome the time-out from NTP Server.

III. Implementation Part

Implementation (Embedding to GENI)
We have given an .xml file which consists the nodes, interfaces and IPs' of them. We have created a slice which named as TPPart1Group26 and added .xml to RSpect then our topology has been created.
We have decided to use Stanford Instageni Site which has available VMs more than we need.
After, we have created a ssh key to connect these VMs locally from terminal. Ssh key must be added to our keys before usage, to do this:
ssh-add id_geni_ssh_rsa
To connect each VM, we connect specific VM from Stanford IntaGENI by ssh(by specifying user-name, host and port).
After establishing the connection to VMs, we need to embed our code. We have written Python scripts for all nodes and use GIT for developing. We have cloned our codes to each VM then run specific scripts on each VMs. Finally all we need to do is running the scripts with paying attention to running source's script at the end(Required since script for source node send message automatically just after it runs.).
In our output we are able to see the time difference between the message's sent time and delivery time, and also we are able to see on which router the message came to the destination.

IV. Experiment Part

After calculating the end-to-end delay, we have calculated the confidence interval for exp0, exp1, exp2 and exp3.
exp0 indicates that results without emulated delay
exp1 indicates that emulated delay with 1ms $\pm$ 5ms
exp2 indicates that emulated delay with 20ms $\pm$ 5ms
exp3 indicates that emulated delay with 60ms $\pm$ 5ms
using tc qdisc add dev "interface_name" root netem delay "delay" "error_rate" on terminal
Then we have used tool to calculate 95% confidence interval from $Confidence\_Interval\_Calculator$ for each of the different communication, where we have inserted our outputs as raw-data then it gives interval with 95% confidence. For each exp the 95% confidence intervals are:
exp0 $=> 118 \pm 1.82$
exp1 $=> 149 \pm 1.84$
exp2 $=> 157 \pm 1.06$
exp3 $=> 252 \pm 2.44$

After that, to plot the graphic we have used math-lab, code is below:

```
>> x = [0,1,20,60]
>> y = [118,149,157,252]
>> err = [1.82,1.84,1.06,2.44]
>> figure
>> errorbar(x,y,err)
>> grid on
>> xlabel('Network Emulation Delay (ms)')
>> ylabel('End to End Delay (ms)')
>> legend('Delay')
```

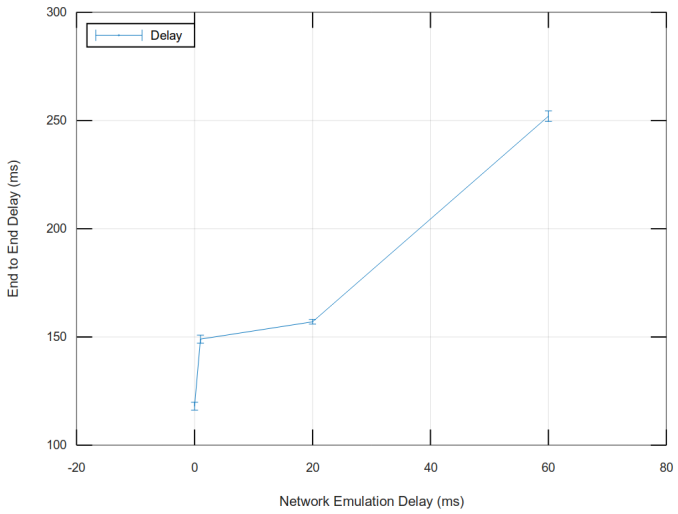

Fig. 2. The graph of Network Emulation Delay vs End-to-End Delay with a 95% confidence interval

The delay in our experiment may be caused because of:

- Our added emulated delay
- Congestion of used network

As we can see in the graph, when we have increased emulated delay, end-to-end delay is also increased. The reason of increase between exp0 and exp1 might because of congestion of network. The increase between exp1-exp2 and exp2-exp3 are proportional to our emulated delay variables. The network can be re-designed according to needs, when we need to make it faster we can use UDP instead of TCP but when reliability comes in TCP is what we need.

V. Conclusion

After we collect the data, we examined the difference between TCP and UDP connection To do that, we calculate time in broker and one of router. While some packets travel faster in UDP, some don't. In theory, UDP is faster because error recovery is not attempted. It is a "best effort" protocol and TCP does error checking and error recovery. Erroneous packets are re-transmitted from the source to the destination. The contradiction may cause, we are sending small sized packets as our packets arrive in msecs or maybe caused from devices.

To conclude, in this project we learned how to create sockets in Python and send packets from one node to another over TCP and UDP and how to calculate end-to-end delay. We also have calculated the confidence interval, which helps us to use results easily with allowable error rate. The network delays are important especially in systems which operates simultaneously to overcome the problems caused by delays in network. Delays are the important factors to be calculated and design the system according to that for computer engineers especially working on huge systems.