Term Project

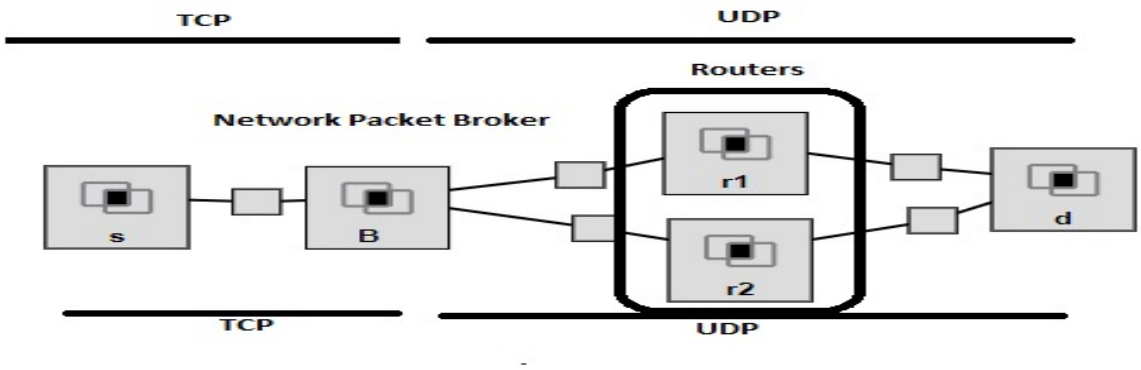
1.Steps

We followed these steps in this assignment:
1.step is that :Getting prepared to use Geni Projects
2.step is that :Understanding Our Topology 3.step is that :Determining how to implement
4.step is that :Writing Python Socket Programming to apply system
5.step is that :Uploading codes into virtual machines

2.About GENI Platform

Before trying to prove geni platform with lab 0 and lab 1 like supposed in assigment, we take our geni account . Geni tutorials aren't enough for us to understand GENI complately.That is why, we watch some videos about GENI on Youtube. Thanks to that, we understand how to produce a topology ,how to create slices and how to be able to control virtual machine in this topology via SSH.

3.About Our topology



There are four kinds of nodes,namely source ,destination, router and a broker.
Source Node: 's' can be considered as a source device which collects sensor measurements and sends this information to the destination node.
Destination Node : 'd' can be considered as a data center which receives sensor measurements from the source over both r1 and r2.
Router Nodes: r1 and r2 can be considered as routers. Both of them transmit data from source to destination simultaneously.
Broker: A broker(B) is a device which has the ability to listen data from source via TCP and sends packets to both routers(r1 and r2) via UDP .
1-) s : Its main aim is to send packet to B via TCP

2-) B : It always listens to s like a server via UDP (This means that,B will be able to receive from s), and store and forward the received messages like a router.It should

also have the ability to get streams of bytes based on TCP and send packets based on UDP by using multiple links.B has to convert TCP signals to UDP datagrams.

3-) r1 : r1 ,which is related to listen link B so as to accept packets from B via UDP, works as a server node.In addition to that, it sends packets to d by using UDP ,which shows that it works as a client node at the same time .

4-) r2 : r2 has the same working principle as r1.

5-) d : Just listens link to r1 and r2 receive packet and give output both from r1 and r2, which means that if we send 'a' string from source, destination gives us ('a' + 'a') as a output for r1 and r2.

The table which is below shows r1 and r2 routing tables.Thanks to that , we can understand where is the next station to send for reaching our packet to one.

Destination	**Send To**
s	B
B	B
r1	-
r2	B
d	d

Table 1: $r1$ Route Table

Destination	**Send To**
s	B
B	B
r1	B
r2	-
d	d

Table 2: $r2$ Routing Table

4.Programming Part

Since, there is no restriction which language is used, we used python for this assignment in programming part. Since Python has more useful modules and its syntax needs much less effort for us.

We used main two modules for socket programming.
Because of any limitation, we use Python as a programming language instead of C++ because it is easier on Python to encode socket programming for us.

We edit our Python Codes in vim editor.

We import socket,time and sys libraries in sTcpClient.py.

We import only socket library for other Python Codes.(dUdpClient.py,r1Udp.py,r2Udp.py)

We import socket and threading for brokerUdp.py.

We cannot synchronize time so we test our time data on local platform. Therefore we send 1000 data and divide total delay by 1000.

We solved a problem, which is file /usr/lib/python2.7/socket.py line 228 in meth by killing process (pykill -9 python)

1.Time : While sending data

2.Socket: While sending message and creating socket, we used this module.

3.Threading: Since we need simultaneous access for broker,we had to use concurrency. That's why while broker listens two socket which one receive packet, to handle this task we used thread function in this module.
We used routing table for finding route in broker.

5.Uploading codes into virtual machines

We connected our nodes on GENI by using SSH. We write our codes on Local and then we implement them on nodes(etc. s..) via Vim Editor

6.Testing communication and delays

We cannot synchronize time so we test our time data on local platform. Therefore we send 1000 data and divide total delay by 1000.