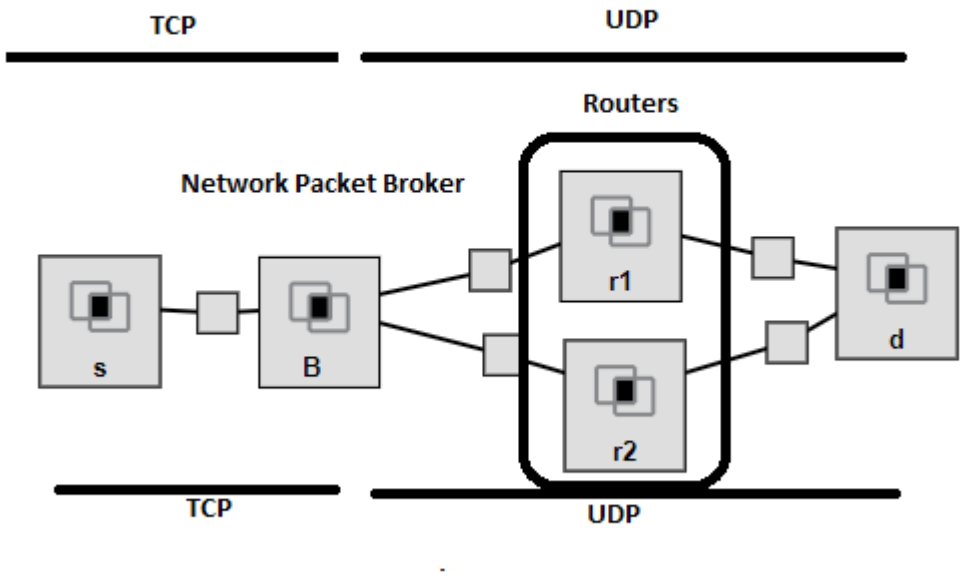
Term Project 1

In this project you are expected to design a network which consists of a source node, a broker (B), routers and a destination node. This project will be divided into two parts. The first part will be a partially "unreliable" network and the second one will be a "reliable and multi-homed" network. Hence, in the first part of this project, assume that we will send some messages (you will create these messages, its up to your design) over an unreliable network, however, in the second part of the project we will send a large file such that we want to transmit intact from source to destination.

A. Topology

The network has the below-specified topology. There are 5 hosts. While you are creating your slices, please use an XML file ("Save as" to download_topology.xml). Please, look at the "Geni tutorials to create a slice with using an XML file". In this XML file, we created the nodes with manually IP addresses. Thus, everybody works on the same topology, with the same interface IP addresses, and with the same configuration.



Source Node: s *will be considered as a source device which collects sensor measurements (***(you will create these messages, its up to your design)** *) and sends this information to the destination node.*

Destination Node : *d will be considered as a data center which receives sensor measurements from the source* **over both r1 and r2.**

Router Nodes: *r1 and r2 will be considered as routers.*

B: It stands for the broker. A broker is a device which has the ability to send packets over multiple paths. it should also have the ability to get streams of bytes based on TCP and send packets based on UDP by using multiple links. It has to convert TCP streams to UDP datagrams, or UDP

datagrams to TCP streams. During the emulation of the applications, this node will be executed **firstly (Only one script will execute and manage its tasks)***, then other nodes will be connected. It always listens to the network like a server, and store and forward the received messages like a router. if it receives a packet that may be a TCP segment or a UDP datagram (it may change), it will send the received packet by considering the network protocol at that time. B node has to manage multiple requests simultaneously, in other words, there may be one or more messages received from different nodes at a time. As a hint, you can take advantage of like a structure consisting of threads.* **Any other implementation and design issues are up to you (packet size,which packets will be sent to r1 and r2, using threads, multiple interfaces etc.).**

B.

In the first part of this project, some nodes employ the User Datagram Protocol (UDP) based socket application, and some of them employ the Transmission Control Protocol (TCP) based socket application. The broker has to implement both of these two application layer protocols that you are expected to develop and deal one/multiple links and send the message (your data and control packets) to two routers as specified in the Topology section. In this assignment, you are going to build and test virtual networks on GENI Platform (https://portal.geni.net/). The protocols will be tested on a specific topology consisting of five (5) hosts. As it can be seen in the topology of the network, TCP based application layer protocol is employed between the nodes source(s) and the broker. UDP based application layer protocol is exploited between the broker and destination (d). The node which should handle TCP and UDP is named as broker. In this topology, r1 and r2 are considered as routers.

1. Specifications

- You are expected to develop **UDP and TCP** socket applications.
- *There are three different types of devices: nodes with TCP or UDP, router nodes, and B.* **At each of the nodes only one script should run.**
- Your source will send this **a list of sensor readings** to the destination. However, The messages should follow the following path: Firstly, The messages should come to the broker as the next hop from the source. Once B starts to get the byte streams, it will packetize the stream. Then B will send the packets to r1 and r2. In the end, the packets will be received by the destination from both r1 and r2. The order, comparison of messages or other implementation details are up to your design. (As a recommendation, in part2 since we will use a file to send from source to destination you can put these sensor messages into a file for part1 and you can read by splitting these messages as chunks and then you can send each of the chunk as sensor messages from source to B.)
- *r1 and r2 should have a routing logic implemented at the application layer. You can create a file including the routing table of r1 and r2 in them so that you can use this routing table in order to route your data or control packets, or you can directly use the next hop address in your routing scripts. In other words, r1 and r2 will be like a router that stores and forwards the data to the next hop.*
- Your codes are expected to include the necessary comments about the functionality of the code segments besides the README file. You are expected to use netem/tc Linux commands for the link configuration. This is a part of this assignment.

- You will learn the usage and use it to configure links as specified below. Please, being aware of how to do such a configuration if you configure more than one parameter by using this command line tool is important.
- **You are expected to plot the following figure:**

Plot a figure that provides the relation of **network emulation delay** and the **end-to-end delay** with a 95% confidence interval for each of the different communication. We will try to see that how end-end delay (form s to d) will change in this network.

For delay, use "normal distribution" as it is defined already.

Experiment 1: 1ms+-5ms
Experiment 2: 20ms+-5ms
Experiment 3: 60ms+-5ms
For the usage of 'tc' command: (**All of these configurations should be applied to all links between Broker and d**)
http://lartc.org/manpages/tc.txt
https://wiki.linuxfoundation.org/networking/netem

***** While calculating the end-to-end delay you can use the following manner (optional you can also use a different approach):**

- There should be a client program that sends a packet from node s to d and measure the end-to-end delay.
- There should be a server program on node d that receive a packet and send back the control feedback (not an obligation) (e.g., ACK or NACK if it is possible).

Please look at the following link for the details and usage :

https://wiki.linuxfoundation.org/networking/netem