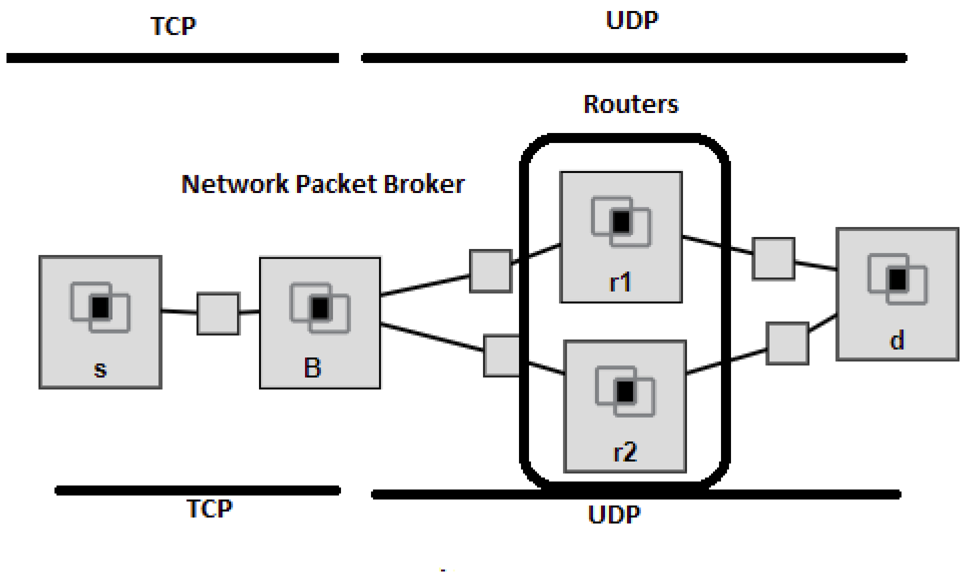
Term Project (Part1 and Part2)

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. Part1

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

2. Reporting

- Use LaTeX to write the report. It should not be longer than 6 pages. **Please use the IEEE Conference LaTeX template.**
- **The report must consist of your design and implementation approach, your methodology, motivation and your experimental results with their explanations and your comments to them. Explain in detail how your broker implementation handles both TCP and UDP, what is the difference between TCP and UDP in this topology in terms of your results, etc..**
- Plot emulated delay versus end-to-end delay graph. Your comments about the change of the graph will also be graded. (What is the cause of the delay, what was expected, what can be changed, etc.)
- Note that for each test, you have to execute your code **many** times and take the mean of your results before calculating end-to-end delay.
- **You also have to provide the code, and “how to run” within a README file. At the top of the README file, you should write ID, name and surname of both group members**

as commented out. ReadMe File should also include all configuration commands step by step, how to synchronize the nodes, how to apply the tc/netem commands etc.. Please explain all of your steps you follow to execute your scripts for each experiment.

3. Rules (Tentative: In the evaluation process, these points may change)

- Graph and the description will be 10 points.
    - Legend, axis labels, captions and units missing : -2 points
    - The description is not specific or not clear enough : -3 points
    - Description has wrong conclusions or missing key points: -5 points
- Report and ReadMe File will be 40 points.
    - LaTeX is not used: -5 points
    - Calculated RTT instead of end-to-end delay -5 points
    - Used wrong parameters: -5 points
    - ReadMe File: 5 points
    - Insufficient comments: 10 points
- The code will be 50 points.
    - Code not working/missing protocols/software engineering principles are not followed: -20 points
    - Not obeying specifications such as (Routing, single scripts running on each node etc.) not followed -20 points
    - Comments on the function of code segments missing -10 points

4. Deadline

The deadline is 2nd of December, 2018 23:55.

5. Submission

Use ODTUCLASS to submit the TP_Part1. There will be two submission steps: one of them is the submission of your implementation files and the other one is your report submission. The first step will be performed under the **TP_Part1 Implementation** and the other one will be under the **TP-1 Report** Assignment. Thus,

**** Submit your codes, latex files, and the README file on ODTUCLASS using the **TP-1 Implementation** Assignment **(TP-1 Implementation Submission)** as a single file named as TP_Part1_##.tar.gz. Then,

**** Submit your report to "**TP-1 Report Submission**" as TP_Part1_##.pdf.

Do not forget the replace ## with your group number.

This assignment is a Turnitin assignment. According to METU regulations, we will tolerate up to 20% of similarity results.

If your result is bigger than 20%, your assignment will be evaluated as zero.

ONLY ONE GROUP MEMBER SHOULD SUBMIT THE HOMEWORK.

6. Extra Comments

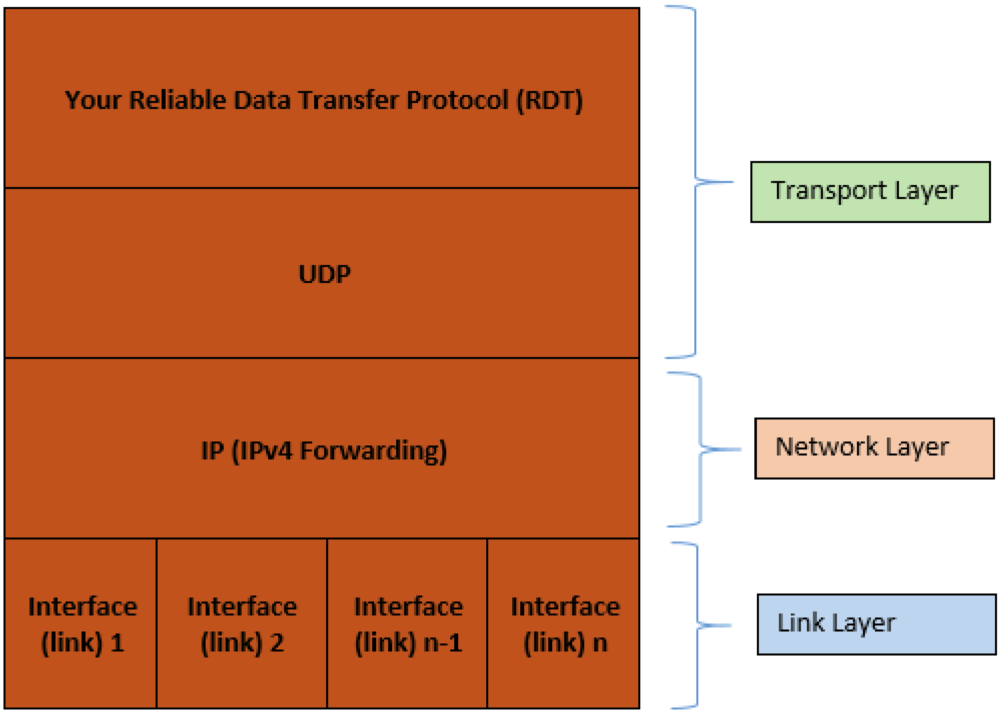
* While plotting the figures, MatLab (Octave, Gnuplot) will ease your job if you can set up the output accordingly.

* Since this is a group homework, the coding part will be same for the members of each group. Sharing the workload is OK, but it is not allowed to separate the coding part & the plotting part completely.

* In case of cheating, the cheating policy of our department will be strictly followed.

C. PART2

In this homework, you are expected to implement a **UDP-based** "**Reliable Data Transfer**" (**RDT**) protocol of **your own (not same as RDT 3.0 etc. in the textbook and literature)** that supports **pipelining and multi-homing**. You will also be expected to **change your network configuration**, in other words, you should modify the routing table of the nodes. Note that your protocol should be implemented by considering your own approach and design. You will concentrate on the specified topology connecting two edges and one broker (**s, B**, and **d**) over disjoint paths. The **source** will be the host **s,** and the **destination** will be the host **d**.



1. Network Configuration

You should be familiar with the **IPv4 addressing and routing mechanism**. You have a source and a destination node in this topology. The other nodes (i.e., routers) should forward the packets to the next hop. You have already done this work by implementing an application in the first assignment. However, you are now expected to make this configuration by modifying the routing tables of each node, namely, routing in the network layer. Please look at the "route" command for the details, and IPV4 routing mechanism implemented in the kernel.

For Instance: When a packet comes to **r1** from **B**, **r1** will forward the packet to **d**. Otherwise, if a control packet (ACK/NACK) comes from **d** to **r1**, **r1** will forward it to **B**.

For this homework, you cannot use any routing script in the routers since they will route the packets to the destination with the routing mechanism.

Moreover, you can give directly the destination IP address as a parameter to your source application. You will run your scripts only in the source, broker B and destination d. You have three terminals for running your codes one of them is the source side, the second one in the broker side, and the other one is the destination side.

2. Specifications

- You will implement and develop your own RDT protocol on top UDP, Thus, in this part of the project instead of UDP, you will use your own reliable protocol to send a large file from s to d. In other words, the unreliable part of the network will then be reliable. Then, the communications will be between TCP and your RDT protocol.
- Now, the broker should handle the links in a reliable fashion.
- Develop your reliable transport protocol that supports multi-homing by using UDP sockets. The above figure shows the stack including on which layer you do your implementation.
- Develop a packet-based protocol.
- You can use any programming language to implement your code, as soon as you explain the usage of it.
- Exploit disjoint links between the broker and destination. We will use the advantage of two links while transfering the file, exploiting these multiple paths will allow us to transfer the file faster than using one path. This part will provide us a multihomed protocol that you are expected to implement.
- Your source will send this large file (exactly 5 MBytes) to the destination.
- The file will be protected with a **checksum** in the assessment. In other words, the input file at the source side, and the transmitted file at the destination side should be exactly the same.
- Design your packet structure that will go into the UDP payload. **The maximum packet sizes of your RDT protocol (header + payload) can be at most 1000 bytes.**
- **Notice that, your protocol should be reusable for any topology or any network configuration based bandwidth,** delay**, and packet size. Show these details in your codes with your comments. Here, your codes will be graded based on only above topology.**
- Your codes must be **executed your source code on host s, and your destination code on host d. Any application for the routers will not be accepted. The aim of the routers is that they just route the packet to the destination and source by means of using routing mechanism (ipv4 forwarding and routing table, not applications).**
- **Run your source code on host s, and your destination code on host d:** You are expected to produce the folowing figures by changing the link properties by using "tc" command. You will change the loss, corruption and reordering property of the links explained below.
- For the usage of 'tc' command: [http://lartc.org/manpages/tc.txt]
  [https://wiki.linuxfoundation.org/networking/netem]

>>> You are expected to plot the following figures by using your own reliable protocol:

1. x-axis: **packet loss percentage (0.5%, 10%, 20%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be applied for all links. Notice that initial configurations should also be applied explained below. In this figure, the results of your RDT protocol should be represented as a line chart or bar graph.

You will present your results with considering all experiments.

```
Initial Configurations for Figure 1:(Please apply it for each corresponded
experiment for all links between B and d)
```

```
tc qdisc change dev [INTERFACE] root netem loss 0.5% corrupt 0% duplicate
0% delay 3 ms reorder 0% 0%

tc qdisc change dev [INTERFACE] root netem loss 10% corrupt 0% duplicate
0% delay 3 ms reorder 0% 0%

tc qdisc change dev [INTERFACE] root netem loss 20% corrupt 0% duplicate
0% delay 3 ms reorder 0% 0%
```

2. x-axis: **corruption percentage (0.2%, 10%, 20%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be the same for all links. Notice that initial configurations should also be applied explained below. In this figure: The results of your RDT protocol should be represented as a line chart. You will present your results with considering all experiments.

```
Initial Configurations for Figure 2:

(Please apply it for each corresponded experiment for all links between B
and d)

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0.2% duplicate
0% delay 3 ms reorder 0% 0%

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 10% duplicate
0% delay 3 ms reorder 0% 0%

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 20% duplicate
0% delay 3 ms reorder 0% 0%
```

3. x-axis:**reordering percentage (1%, 10%, 35%)**, y-axis: **file transfer time** with 95% confidence intervals. Keep the delay the same as 3 ms. Packet loss will be the same for all links. Notice that initial configurations should also be applied explained below. In this figure: The results of your RDT protocol should be represented as a line chart. You will present your results with considering all experiments.

```
Initial Configurations for Figure 3:

(Please apply it for each corresponded experiment for all links between B
and d)

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
delay 3 ms reorder 1% 50%

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
delay 3 ms reorder 10% 50%

tc qdisc change dev [INTERFACE] root netem loss 0% corrupt 0% duplicate 0%
delay 3 ms reorder 35% 50%
```

Notice that, for plotting the **confidence interva**l you will have to repeat each experiment multiple times, say *n*. The margin of error less must be than 2.5%. The z-score is 1.96 for 95% confidence.

We expect that the standard error (deviation) will become smaller as you increase *n*. You will have to compute the value of *n* in your experiments based on the standard error you achieve.

3. Reports

- Use latex (you will use the same template) to write the report. It should not be longer than 8-9 pages.
- **Please write your problem & solution, methodology to overview of your codes, which mechanisms use to provide** reliability **and** how, **scripts and your approach. In your reports try to analyze and demonstrate your results.**
- In your reports, you should explain all of the developments and implementations, strategies, approaches, their reasons for your own protocol. For instance: How do you provide multihoming, pipelening (if you use go-back-n or selective repeat or your method etc. Explain them), which mechanisms you use to implement RDT(checksum, retransmission, ack/nack mechanism, sequence number, buffering etc.) and which of such mechanisms you use to handle the loss, corruption and reordering, and how they effect your implementation.
- Any design issue or algorithm selection are based on your decisions as long as it provides the transmission of a large file between two hosts in a reliable and pipelining multihomed approach. It is important that you should explain of your solutions to the problems clearly.
- Please also be sure that you are expected to make a read me file. For its content refer the assingment text. Your configurations step by step, synchronization of the nodes, your first ip tables and your ip table modifications then your new tables, all of the tc/netem usages etc. In addition to your how to execute your scripts for each experiment and for each step.
- Please put your coding comments into all of your scripts. All of them should be explained the reasons for the usage.
- After obtaining the results and plotting these figures; report the figures and write your comments about the figures. You are expected to write at least one paragraph per figure. Ensure that the figures are self-explanatory: labels, captions, and units are clearly stated in the figures. Use the paragraph to draw the attention to the most significant points of the figure, give specifics but do not repeat details. Ensure image clarity. Use legends.

You also have to provide the code, and “how to run” within a README file. At the top of the README file, you should write ID, name, and surname of both group members as commented out. ReadMe File should also include all configuration commands step by step (from time synchronization to using routing command). For the network configuration part show your IP tables, your used commands to make the configuration and other settings. Please explain all of your steps you follow to execute your scripts for each experiment.

4. Submission

Use ODTUCLASS to submit the homework. There will be two submission steps: one of them is the submission of your implementation files and the other one is your report submission. The first step will be performed under this Assignment and the other one will be under the Turnitin Assignment. Thus,

**** Submit your codes, latex files, and README file to **this assignment (TP-2 Implementation Submission)** as a single file named as TP_part2_##.tar.gz. Then,

**** Submit your report to "**TP-2 Report Submission**" as Tp_part2_##.pdf.

Do not forget the replace ## with your group number.

This assignment is a Turnitin assignment. According to METU regulations, we will tolerate up to 20% of similarity results.

If your result is bigger than 20%, your assignment will be evaluated as zero.

ONLY ONE GROUP MEMBER SHOULD SUBMIT THE HOMEWORK.

Your source codes will be checked with the codes with different resources in terms of academic dishonesty.

5. Suggestions

* While plotting the figures, MatLab, Octave or Gnuplot will ease your job if you can set up the output accordingly.

* Each group member may get different grades. Sharing the workload is OK, but do not separate the coding part & the reporting part completely. **Express clearly in the document who has done what!**

* You can use any programming language.

* You can use any idea or approach unless it is specified in this document.

6. Evaluation (TENTATIVELY)

1- Reliable Data Transfer protocol implementation 25 **points**. In detail, 25 points for your RDT protocol with **a proper checksum** for the above topology for experiments. In other words, your input file should be transmitted to the destination properly. If your code is successful regarding with **multi-homing and pipelining**, you will get **25,12 points respectively**. These points also includes the explanation of your mechanisms, design etc. in your reports.

2- Network configuration (your codes, commands, and IP Routing tables etc.) **10 points**.

3- Each figure and the associated paragraph will deserve 6 **points**.
3.1 Legend, axis labels, caption and units missing: -2 points
3.2 Paragraph is not specific or repeats details: -4 points
3.3 The paragraph derives wrong conclusions or not explanatory reflecting key learning points -4 points

Therefore, there are 3 figures: 3 x 6: **18 points**

4- **Latex** and **clear ReadMe File 10 points.**

7. Deadline

The deadline is 4th of January, 2019 23:55.