Term Project (Part1 and Part2) - Description

In this project, you are expected to construct a network that consists of 5 nodes, which are connected to each other. This project will be divided into two parts. The first part will be a partial network discovery step (you will find link costs, you know IP addresses, the topology, and existing nodes) by implementing a simple User Datagram Protocol (UDP) in addition to one experiment, and in the second part, the specified network will use your own "reliable and multi-homed" protocol to transmit a large file. In the first part of this project, assume that you will send your own messages among the nodes, 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

In this assignment, you are going to build and test virtual networks on the GENI platform (https://portal.geni.net/). The protocols will be tested on the below specific topology consisting of five (5) hosts.

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.

Initial Configurations:

After the reserving process of the topology by using the GENI platform is completed, in order to accomplish the topology creation, you need some initial configurations: (For this purpose) please, run the linked shell scripts "configureR1.sh" and "configureR2.sh" on only the node "r1" and "r2" by connecting the corresponding nodes.

B. Part1

In the first part of this project, nodes employ the User Datagram Protocol (UDP)-based socket application.

1. Specifications

- You are expected to develop a **UDP** socket application.
- At each of the nodes, only one script should run. Each node should have a client/server application. Each node should receive and send messages.
- You can use any programming language to implement your code, as soon as you explain the usage of it.
- ***All nodes*** *will implement a client/server application that sends and receives* ***multiple messages at the same time****.*
- **Each node should handle multiple requests.**
- Our aim is to find the link costs by **considering the RTT values between each hop (node)**.
- **While estimating the RTT values you should collect and average the multiple measurements. Therefore, assume that you send 1000 discovery messages and find the RTT values by averaging them.**
- **The scripts for each node should be executed once only, and until they finish their messaging between each other to discover the costs of the links, the scripts should run. After terminating the messaging, the scripts on the nodes (r1, r2, and r3) should save the costs of the links between its neighbors to an output file named (link_costs.txt) (can be separated documents). Each of the nodes (r1, r2, and r3) should find and save the link costs values between the neighbors. You can choose and implement one of the following scenarios or you can propose new scenario: The costs of the links (s-r1, r1-r2, r1-d) will be estimated and saved by r1, the costs of the links (s-r3, r2-r3, r3-d) will be estimated and saved by r3, and finally, the costs of the links (s-r2, r2-d) will be estimated and saved by r2; or the other scenario: the costs of the links (s-r1, r1-d) will be estimated and saved by r1, the costs of the links (s-r3, r3-d) will be estimated and saved by r3, and finally, the costs of the links (s-r2,r1-r2, r2-r3, r2-d) will be estimated and saved by r2. Each node has to be able to send and receive messages at the same time (multiple requests) from its neighbors depending on these scenarios.**
- You will use an application-level routing logic for the nodes.
- The messaging approach is up to your design, as much as you provide the specs above (**Each node has to be able to send and receive messages at the same time (multiple requests) from its neighbors depending on these scenarios.**). For this purpose, as different examples, you can choose the node "s" as an initiator of the messages or other nodes can be chosen. For example, this node starts to send some messages and a copy of the messages over each of the available links, then you will store and send these packages hop-by-hop to determine the RTT values of each link at the same time. Each of the nodes (r1, r2, and r3) should calculate the link costs for each neighbor at the same time by obeying the specifications above. Note that, any node should not wait for others to finish its 1000 messages. The messages should be directly stored and send when it is received. As another messaging scenario without waiting for an initiator, you can run firstly the script in r2 and then the others (r1, r3, d, and s) respectively, each scripts then will send and receive messages after they start to run. These are the possible scenarios you can use. However, you should consider

the above specifications about the execution of the scripts which they should run only once, and handle multiple requests at the same time. Any other implementation approach or scenario obeying the rules is up to your design and acceptable. You should explain all the details in your reports.

- After finding the shortest-path, you will implement and modify your scripts in addition to a routing logic based on this shortest path for an experiment. The node s, r1,r2,r3, and d should have a routing logic implemented at the application layer. You can create a file including the routing table of each node locally so that you can use this routing table in order to route your data or control packets.
- 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 characterize 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.

2. Link Costs

You are expected to find the costs of all links with **round-trip time (RTT)** as the cost metric. You are expected to find the RTT values of each link by sending some discovery messages multiple times (such as 1000 messages) and then take their average to calculate RTTs for each link. The approach for the transmission of discovery messages is specified above.

3. Finding the Shortest Path using the Dijkstra Algorithm

After calculating the costs of all links, you can find the shortest path from the **source node "s"** to t**he destination node "d"**. You will find the shortest path based on the Dijkstra Algorithm. You are expected to create a table that shows all available paths from one node to another in addition to their costs and using this table you can find the shortest path. For the algorithm, the initial node will be "s". You should find the shortest path between the node "s" and "d". In your reports, you should present this table and all the iterations of your Dijkstra algorithm in this topology. In this part of the term project, you are expected to use the Dijkstra algorithm without using any implementation. You will make these calculations manually to find the shortest path, and present them in your reports. Therefore, the table and the algorithm's steps, and explanations are enough for this part.

4. Experiment

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 how end-end delay (form s to d) will change in this network. You are expected to send messages by obeying to the shortest path that you have already found.

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

Experiment 1: 20ms+-5ms
Experiment 2: 40ms+-5ms
Experiment 3: 50ms+-5ms
For the usage of 'tc' command: (**All of these configurations should be applied to all links of the following three nodes s, r3 and d.**)

***** Link Characterization**

You should characterize the links based on the following link configuration using emulation delay. Apply, all these links before starting the next step.

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.
- There should be a server program on node d that receives a packet, and measure the end-to-end delay and it can send back the control feedback (not an obligation) (e.g., ACK or NACK if it is possible).
- You need the synchronization of the nodes, you can use NTP etc. for this part, This is also the part of your homework.

4.1. Routing Logic Implementation for the Experiment

You are expected to implement an application-level routing logic. You can keep the routing information in a file, and update it for any requirement. The routing information may be different for the link-cost discovery and the experiment. Therefore, you are expected to provide the routing information which should be proper to the specification in your assignment submission. In other words, put your scripts into different folders: a folder called "discoveryScripts", and the other folder "experimentScripts".

Up to this point, while you are sending some messages between the nodes, you find the costs of all links and calculate the shortest-path of your topology where the starting point is your source node "s", and the final node is the destination node "d". Now, you will conduct some experiments using the found shortest-path of this topology. For this purpose, you will send your messages only over the shortest path. For this experiment, you can write different scripts and modify your client/server scripts and routing information to send a list of messages from the node "s" to the destination point "d". Please put your scripts for discovery and the scripts for this experiment to your submission folder separately, if they are different. If you can handle using the same script you can inform in your reports and you can use the same scripts.

5. Reporting

- **Use LaTeX to write the report. It should not be less than 4 and more 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 source, router implementations, and destination implementation handle UDP, and multiple links at the same time, your approach to finding the costs of all links, your table displaying the nodes and their costs, your calculations for Dijkstra Algorithm, your end-to-end delay calculations, differences between end-to-end delay and RTT calculations, your routing implementations, modifications, etc.**
- Plot emulated delay versus end-to-end delay graph. Your comments about the change in 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 in READme file instead of inside of your report.

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

- The 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: 5 points
    - Proper end-to-end delay for the experiment 5 points
    - Used correct parameters for emulation delay: 5 points
    - ReadMe File: 5 points
    - Discussions: 20 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

6. Deadline

The deadline is the 22nd of November 2019 at 23:55.

7. 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,

**** Put your scripts into different folders for the discovery part and the experiment part: a folder called "discoveryScripts", and the other folder "experimentScripts"

**** 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.

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