CS358 2023 Assignment 4 - Distance Vector Routing Protocol

Mayank Agarwal

Deadline: 30 Oct, 2023. 23:59. All of your git repos shall be pulled after that. That will be the version which will be checked.

Warning: Sharing is Caring is good for cat videos. Sharing of program may lead to plagiarism and would effect in 0 to both.

Doubts: All Doubts relating to CS358 2023 Assignment shall be posted on Google Form
https://docs.google.com/forms/d/e/1FAIpQLSdVylWbeelAnQKYDCfqoyxcqSwm_FiVJDbzszV2Mpg-zZCfU
viewform?usp=sf_link

I will respond to the queries here:
https://docs.google.com/spreadsheets/d/1jHhkLLGrRSToYZH857D6hM3SY8Bi0LoTtqDZ-rNrvGw/
edit?usp=sharing

Please avoid email / wa / dm

So common doubts can be solved and we shall be able to keep track in an organized manner.

Pull This Git Repo - https://github.com/cs358/308_CS359 and copy the tut04 to your repo folder.

Git Requirements: At least 3 git commits should be there with meaningful comments (at least 4 words)

Task:

In this assignment, you will be implementing the "Distance Vector Routing" protocol. Since it follows a decentralized approach, you will be implementing this algorithm via multithreading in C/C++ (threading is mandatory). Each thread will represent a router instance. The threads (routers) can communicate through a dedicated queue.

For the following sample topology,

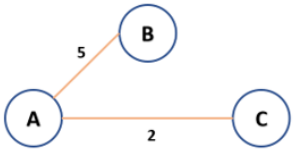

Figure 1: Sample Network Topology

the program should parse the following input to understand the topology of the network:

Assume input from topology.txt located in the same folder as the code.

3
A B C
A B 5
A C 2
END

where:

- The first line represents the number of routers present in the network.
- The second line represents the fixed name (interface) of the router.
- The third line like "A B 5" represents the cost of link connecting nodes A and B. Similar semantics is followed until the last line.
- The last line represents the end of file (denoted by "END").

Output and implementation:

The code should initially check that the input topology is connected. By connected we mean that it should not have a disconnection between vertices. An example of dis-connected graph is shown below. If the input is a dis-connected graph, a message should pop up to the user saying that its a dis-connected graph and the program should quit.

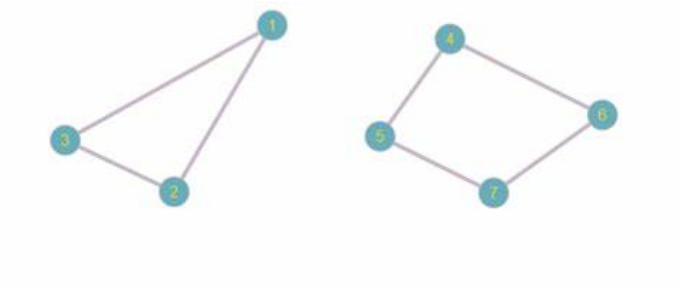

Figure 2: Sample dis-connected graph

The output should initially display the routing tables of each router. After 3 seconds, each router should forward it's routing table to the adjacent routers using a shared queue. The routers should now update their table entries using the Bellman Ford equation. The updated routing tables of each router should be displayed on the terminal.

Note that each time the routing tables are displayed, the iteration number should be also displayed along with it. Also, those entries of the table that were updated in the latest computation round should be marked with an asterisk (*).

After each computation, the routers should wait for 2 seconds and then forward the tables. Since each router knows the number of adjacent routers, each router will wait to receive the table from ALL the adjacent routers and only then perform computation.

The output to be displayed should be very intuitive and must clearly display the entries of each router after every iteration.

It should be noted that the computations should be performed in a decentralized manner in each individual thread. The code should be well documented.

PS: The code should be generic in-order to work on any input topology.txt. For checking we shall have a complex graph and the code should be able to handle it.