CSE 4344 LAB 2 – Distance Vector Routing Fall 2020

Objectives:

1. Simulate Distance Vector Routing
2. Continued work with IDEs and GUIs

Due: Nov. 15, 11:59:59 pm Difficulty: Easy

These will be individual projects. For Lab 2 the same development rules apply as for lab 1. You can use either Java or Python as the programming language.

You are to implement a simulation of Distance Vector routing. We will work with routers as nodes and links as edges, ignoring the fact that routers actually work with network addresses, not router numbers. You will read a plain text input file which will describe the network. Each line in the file will describe a link. The line will have 3 numbers, the numbers of the two nodes followed by the cost of the link. There will be a maximum of 6 nodes and a maximum of 4 links connecting to any one node. For example, using numbers for the nodes instead of letters, the network in figure 4.6 in the text might look like this:

```
1 2 7
2 3 1
1 5 1
4 5 2
2 5 8
4 3 2
```

You should have a GUI implementation that allows the user to see the Distance Vector table and the routing table in each node. It might be that the easiest way to do this is to have (up to) 6 copies of a “node” program running and have a master program that fed the appropriate information to each node. The nodes and the master can connect with sockets when they need to. If you can show the information clearly in a single window then having multiple programs is not necessary.

You should be able to change the input file to be read by the program.

You should be able to run the simulation in a controlled way so that the user can observe the changes in the tables as the nodes run each step in the algorithm. You can assume that all the nodes exchange tables at the same time and then make one iteration with the new information. The system should be able to detect when the algorithm reaches a stable state. i.e., the nodes are not getting any new information. It should indicate this on the GUI. The GUI should also display the number of cycles that the system takes to reach a stable state. Record this information when the stable state is reached and include it in your writeup.

You should also include an option (button or check box) that allows the simulation to run without intervention and display a total time until the algorithm reaches a stable state. i.e., the nodes are not getting any new information. You should record this time for your writeup. To make this more interesting, you might want to suppress updating the display of the tables as they change until after the system reaches a steady state and the time has been recorded.

You should be able to adjust the cost of any link in the network. You should simulate a line failure by setting the cost of a link to infinity. You should then run this network **from the state it was in.** Run it once in the original single step mode so you can see the tables change. Observe the algorithm's progress and record your observation. Then run it again without the intervention and record the running time. Infinity is 16.

You should then simulate a line repair by setting the cost back to what it was originally. You should then make the same runs and observations as in the previous step.

Write-up:

Your write-up should follow the same rules as with lab 1. In addition, the writeup should include the observations discussed above. In each case, include a comment – at least one sentence, on what you conclude from the observation.

Submission:

Your submission should follow the same rules as with lab 1.

Grading:

Points element:

05	Select the input file
10	Initial link state tables set up in GUI
10	System detects a stable state
20	Runs algorithm correctly in single step mode
10	Runs algorithm without stopping – displays time
10	Change a link cost and run from the previous state
30	Writeup – include all elements discussed above
05	Comments in code

Your comments should follow the same rules as with lab 1.

Deductions for failing to follow directions will follow the same rules as with lab 1.

Important Note:

The same rules about collusion and plagiarism apply as with lab 1.