Alex Merino

CSE 3231

12/3/2024

Running Instructions

If this file is being read, it is assumed that the submission folder has already been unzipped. There should now be three Python (.py) files, a *"topology-1.txt"* file, and a *"topology-1"* folder in the current directory alongside this file. Inside the *"topology-1"* folder are the outputs of the *"topology-1.txt"* folder (separated into subdirectories by algorithm), which was asked for as a deliverable in the assignment instructions.

To run the program, open your CLI/Terminal and run the command

`python main.py {inputFile}`

Where the input file must be a .txt file in the same directory as the .py files. Depending on the version of Python/ system the code is being run on, replacing *python* with *python3* might be necessary. After the command is run, both algorithms will run and save output files to a folder corresponding with the name of the .txt file.

Each time the command above is called and the program is run, if there are files already in the folder with the same name as the input file, they will be deleted then replaced while the algorithms are running. The program requires three Python packages, *os, sys,* and *glob,* which are all standard Python modules and do not require installation if Python is already installed. All other code/functions were created, documented, and tested by myself.

Summary of Code

(All code is documented in the .py files, along with logic behind algorithms)

When the program is called, it takes the input file and sends it to a function which sets up the Link State Protocol. The file is read and parsed, then passed through a for loop. Each time there is a connection with a different time, Dijkstra's algorithm is run, starting at each node in the graph, and is then written to file. After going through all inputs, Dijkstra's is run once more to take into account the last set of updated connections.

After Link State is called, the Distance Vector function is called, which reads and parses the file the same as Link State. Then a while loop is run where the max time is 100 and the graph has been updated in the past five iterations. In the loop, if there are new connections at the current time step then save it to an array. Then check if it is a calculate step or a share step (even or odd), and if it is a calculate step, then update the distances from a node to other nodes in graph based on information from neighbors. If it is a share step, then add the new connections to the graph (connections from current time and time from one step before since nodes cannot be added while completing a calculation step). After adding new connections, the nodes then share their topology of the graph with all their neighbors. A check is done to see if the graph has updated in the current step, which resets patience to 5 or decreases it by 1.

For SPF output, each time stamp for each node is organized by distance, where the closest node to the node specified in the file is the highest, and the farthest is the lowest (like the example output file given). For DV files, each time stamp for all nodes are printed in alphabetical order.