Implementation of Distance Vector Routing Protocol

By

Sadhana Seetamsetty (800904075)

Nilay Pendharkar (800900808)

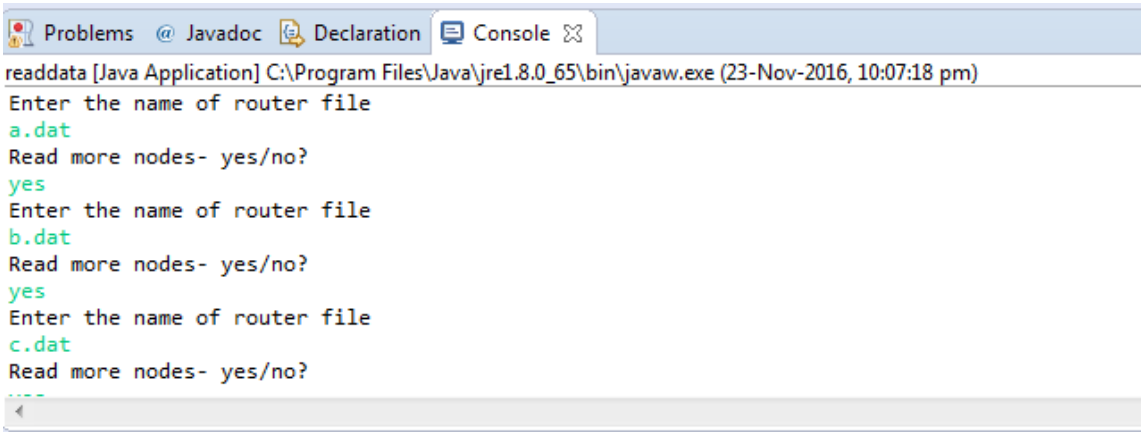
Implementation of the protocol:

The program takes the node file name as input. The file names are stored (so they can be automatically read for updating the routing table every 15 seconds).The node names and the corresponding neighbouring nodes and link costs (read from the file) are stored in a nested hashmap. The initial routing table is populated with the values from hashmaps. After reading all the initial costs into the routing table the iteration begins. The iteration starts off with the first row/node. It reads rest of the rows/nodes in a loop and updates the distances in its nodes. Next it goes to the second row and so on until the last row/node. The program is designed to wait for 15 seconds and then update the table again by reading the files and following the above mentioned steps again. Whenever there is a change in link cost, the routing table is automatically updated. The program is continuously running. The files should be present in the project directory i.e C://Users/….dvrp/a.dat .They should not be present in any sub-directory.

Running the program:

The program prompts the user to enter the name of node file name. After you enter the node file name you will be asked if you want to input more nodes. The file name should be of format **a.dat, c.dat** etc. The nodes can be entered in any order. If you want to add more nodes enter **yes** to the question **Read more nodes-yes/no?** else enter **no**. After the final node has been inputted and there are no more nodes to add, the program will calculate the distance vectors using the bellman-ford algorithm and will print them out. The distance vectors will be printed out for every 15 seconds. Every 15 seconds, the files are read again automatically, user doesn't have to re-enter the files again. Whenever there is a link cost change the program will automatically read the files like as usual and update the routing table and display it.

Input:



Output: