**Implementing Distance Vector Routing Protocol**

**Contributions:**

**Advaith Auron Suresh**

**and**

**Praveen Ganapathy Narasimhan**

INTRODUCTION:

The program implements a variation of distance vector routing algorithm where information is forwarded from each router in a 15 second interval. This helps in retrieving lost information in the following cycle thus making it a very efficient protocol. Once we build a graph of available nodes with its set of link costs, the program would be able display the shortest path from each node to every other node to which it has a defined path or route.

PROGRAM FLOW:

Each program reads three arguments from the command line indicating the port number, the input file name and the node for which the shortest path is to be displayed.

Once the inputs have been successfully received, a graph is recursively built the information obtained.

A timer is triggered in such a way that the adjacent node information, adjacent costs and lowest path cost are updated every 15 seconds.

While the timer thread is separately executed, the program waits to receive information from other ports (which are other nodes). Updating information received from other nodes in this way, we build a graph and display the shortest path from each node continually.

# PROGRAM DESIGN:

Information such as nodes, adjacent nodes and their costs, shortest path length from each node and hops are stored in hash maps which take key value pairs making it easy to relate and manipulate accordingly. Timestamps have been maintained to periodically check and update the program based on input files.

A timer has been implemented and will trigger an event after each 15 second interval. Information stored in maps are sent across sockets as bytes and parsed accordingly to retrieve the data. Poisoned reverse is identified using the next hop information.

A counter has been maintained to keep track of the number of rounds each node has been processed. A file is created to store port numbers.

This file will keep track of different port numbers and writes into the file for each port number from which data is received. The file is repeatedly read to obtain port information used to determine the shortest path.

EXECUTION INSTRUCTIONS:

* Place the DistanceVector.java file along with your “node information input files” (ex: a.dat, b.dat, c.dat) in one folder (convenience of testing).
* Once your java path variables are set, open command prompt in the folder containing the java file and compile it. (Ex: javac DistanceVector.java)
* Once compiled, execute the file using a command in the following format:
  + **Java DistanceVector “PortNumber” “Input File” “Node Name”** (Ex: java DistanceVector 5000 a.txt a)
* Make sure that the input file is of the following format:

3

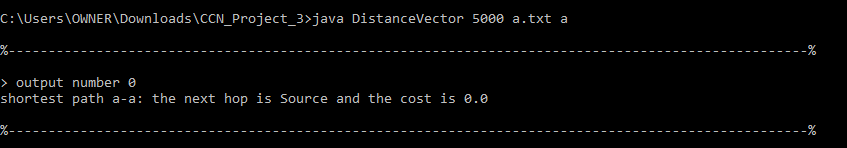
b 5.0

c 4.0

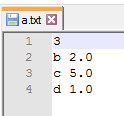
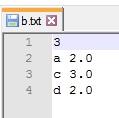
d 3.0

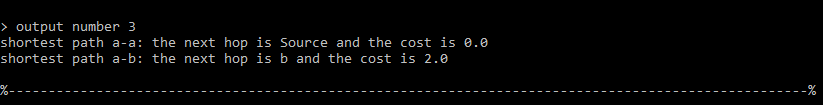
where the first line gives the number of links and the others are nodes and their link costs

* This will execute the file and display a console output like the following format:

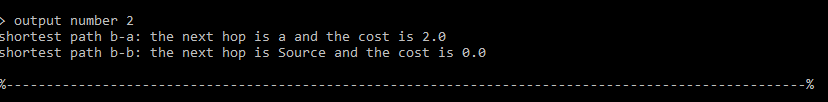


* Open another command prompt in the same folder but for a different node such as b. The command should look like java DistanceVector 5001 b.txt b. Once this is simultaneously run, both the programs would receive information from the indefinite listening functionality placed in each program. The output from both the console for each following input file would be like:



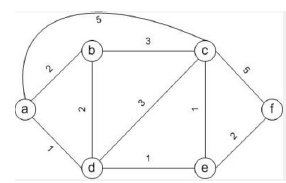
At node a



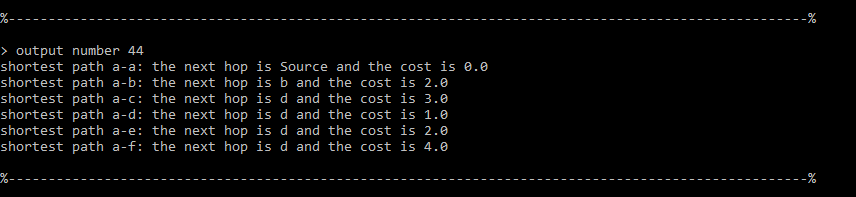
At node b

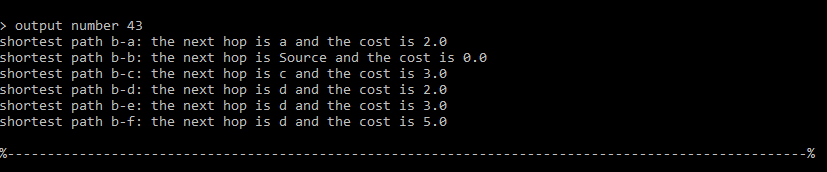
* Similar to running programs for node a and b, run programs for all the nodes in the graph and check if the short distance is calculated for each node.

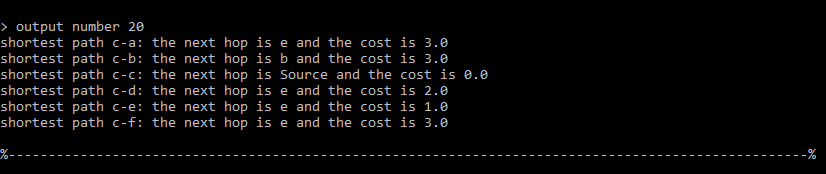
Using the diagram given in the problem statement, we have designed input values according to the following figure,

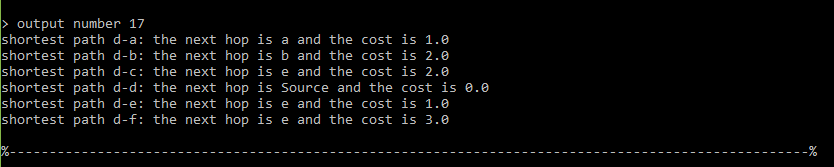


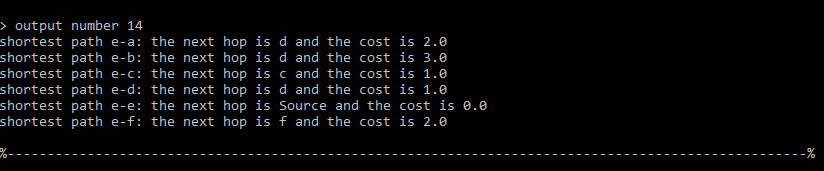
* Once all the nodes are up and running, the program calculates the shortest path and it normalizes after a few iterations. The following are the outputs for node a through f:

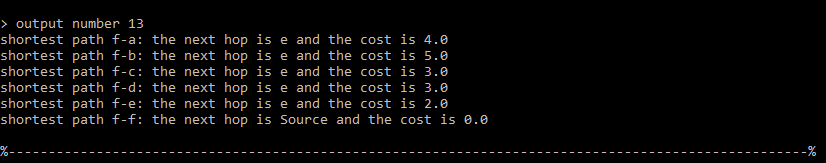












* The screenshots indicate the shortest path from each node to every other reachable node.

# LINK CHANGE COST:

To perform and obtain the link change cost, we will have to modify the input file. While the program is in execution, open one of the input files (say, a.txt) and modify one of the values (say, change c 3.0 to c 5.0 in a’s input file a.txt) and save it. You should also make the other cost in c.txt to indicate the same value (i.e a 🡪 c is the same as c 🡪 a). So, if c 5.0 is defined in a.txt, then a 5.0 should be indicated in c.txt to avoid any undefined behavior. Poisoned Reverse has been implemented in the program where the longer path from a node is identified and made undesirable.