Project 2 Write-Up

Program Design and Explanation

The project consists of four classes, RTableEntry, RoutingTable, RouterStarter and RouterSender.

RtableEntry

RTableEntry is a serializable class. Each RTableEntry object holds one row of the Routing Table.
The variables are:

- String source that holds the name of the source router
- String dest holds the name of the destination router
- String nextHop holds the name of the next router in the path
- double cost holds the cost of the path between the source and destination router

The methods are:

- Getters and Setters for all the variables
- Constructor RTableEntry(String source, String dest, String nextHop, Double cost) that initializes each row in the Routing Table

RoutingTable

RoutingTable is a serializable class and each object is one router.
The variables are:

- ArrayList<RTableEntry> table holds the routing table with each RTableEntry object as an element.
- double infinity holds the infinity value, which has been set to 16 to avoid count to infinity problem
- HashMap<String, Double> originals holds the names of destination routers that are immediate neighbors of the router and their respective costs from this router. This variable doesn't change once it has been read into so only has the names and costs of the immediate neighbors of the router.
- ArrayList<String> immediateNeighbors is an array list of router names that are immediate neighbors of the router this object belongs to
- String routerName holds the name of this object router
- BufferedReader brObj is used to read read files
- String fileName is the name of the file for this object router

The methods are:

- Getters and Setters for the variables routerName, immediateNeighbors and table
- No argument constructor and a copy constructor
- constructInitialTable takes as input the name of the binary data file passed as a command line argument and populates the variable table with

the immediate neighbor entries that are retrieved from this file. It also updates the variables immediate neighbors and originals.

- updateRouterTable takes as input two RoutingTable objects ownTable and receivedTable which are the this routers object and some other routers object that was received by the current router.
  The method first checks for new neighbor entries in the received object and adds them to the own routers object. The cost of these neighbors is set to infinity(16 in this case) and nextHop is set to "-". It updates both table and immediateNeighbors.
  Next it copies the received RoutingTable object into an new object . This object is modified for later comparisn. The distance between this router and the router object was received from is found and added to all the cost values in this new objects table. Next the nextHop entries for all the rows are changed to the routers name that this object was copied from, i.e., the name of the router the other object was received from.
  Then the two update problem is fixed by changing the cost of all the entries whose nextHop value is the name of the receiving router to big finite number.
  This modified table and the own table of the router are compared and the routers own table is updated according to the distance vector routing algorithm.
- printTable takes as input any ArrayList<RTableEntry> variable and prints it out as can be seen in the output screenshots.
- checkLinkCostChange reads the file again for any link cost changes and makes necessary changes. It reads the binary data file, creates another RoutingTable object and adds the values from the file to it. This object is then compared with the original cost values to see if there is any change in cost. If there is a change the cost for that destination, it is updated to the new cost in the original ArrayList<RTableEntry> table and the cost value for HashMap<String, Double> originals is also changed. The nextHop is table variable is also changed to the name of this destination router.

RouterSender and RouterStarter send and receive the UDP datagrams respectively.
RouterStarter uses a MulticastSocket which is added to a multicast group with IP address "224.0.0.251". So any UDP datagrams sent to this address are also received by the MulticastSocket. The MulticastSocket is bound with the port number passed at command line. Everytime a node is run, they initialize a new socket and each is bound with the same port number and added to the same multicast group. So all nodes receive the packets sent out by all the nodes. RouterSender sends UDP datagrams to the multicast IP address "224.0.0.251" and the port number passed at command line when the program is run.

RouterStarter

RouterStarter contains the main method for the project.
The methods are:

- main method first reads the binary data file and constructs the routing table. it then calls the contructor of RouterSender to initialize the sending out process. It the intializes a MulticastSocket which is added to a multicast group with IP address "224.0.0.251". So any UDP datagrams sent to this address are also received by the MulticastSocket. The MulticastSocket is bound with the port number passed at command line. Everytime a node is run, they initialize a new socket and each is bound with the same port number and added to the same multicast group. So all nodes receive the packets sent out by all the nodes.
  When a new datagram is received, the method checks if the router that this object came from is an immediate neighbor, if yes, it uses the datagram to update its routing table else doesn't use it.

RouterSender

The variables are:

- DatagramSocket outSocket, RoutingTable tableBeingSent, int PORT and DatagramPacket outPacket are the datagram socket, routing table being sent out, port number to which the datagram packet is being sent and datagram packet being sent out.

The methods are:

- run sends out data every 15 seconds. It first checks for any link cost changes, then sends the RoutingTable object of the node to the multicast group and prints what it sends on the terminal.

Programming Language & Compiler Version

Java & JavaSE-1.8

Steps to execute the program

- Open cmd/terminal and check the java version by using: java –version
- If the version is desirable run the command: javac –version
- To execute class RouterStarter, run the commands:
  cd <directory-of-files>
  javac *.java
  cd <one-level-above-directory-of-files>
  java <directory-name>.RouterStarter <port-number> <file-path>

  Example:

```
cd /Users/gazalchawla/Documents/workspace1/Try1/src/
project2
javac *.java
cd /Users/gazalchawla/Documents/workspace1/Try1/src/
java project2.RouterStarter 8888 /Users/gazalchawla/
Desktop/a.dat
```

- Do the above for all nodes and **remember that all the nodes use the same port number**.

ScreenShot for one case