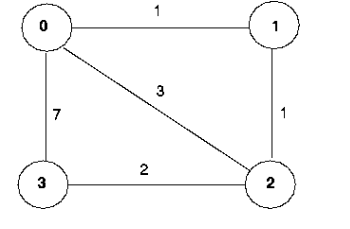
Implemented a set of procedures for the distributed asynchronous distance vector routing for the network shown below:



My programming IDE was eclipse: <http://www.eclipse.org> and my program consists of many .java files namely:

* Entity0.java
* Entity1.java
* Entity2.java
* Entity3.java
* NetworkSimulator.java
* Event.java
* Packet.java
* EventList.java
* EventListImpl.java
* Project.java

**Algorithm: {BELLMAN FORD ALGORITHM}**

The assumptions for distance-vector routing are:

* Each node knows the cost of links of each of its directly connected neighbors.
* Every node sends the configuration message to its directly connected neighbors containing its own distance table.
* Every node will read and update its distance table with cost and send the updated information to all nodes in the network. Repeat the exchanging until no more information between the neighbors.
* Consider Node X that is interested in sending the information to Node Y through directly connected Node Z. Node X distance table entry, Dx(Y) is sum cost of direct-one hop link between X and Z , c(X,Z) and cost of link from Node Z to Node Y, minimum-cost (shortest path) is selected from all X to Y.

Dx (Y) = min{c(X, Z) + Dz (Y)}, for all paths from X to Y.

Hence, whenever a node computes a new minimum cost to some destination, it must inform its neighbors of this new minimum cost.

**Implementation:**

I have only modified and included **procedures** and **constructors** in the 4 java files i,e Entity0.java;Entity1.java;Entity2.java;Entity3.java.

***Constructors***:

Initializing the distance table to reflect the direct costs to the neighboring nodes. The distance table is sent to its directly-connected neighbors about the costs of its minimum cost paths to all other network nodes. This minimum cost information is send to neighboring nodes in a routing packet via the toLayer2() method of Network Simulator.

***Update*()**:

Is invoked by the Simulator when the arrival packet destination belongs to it .The values it receives in a routing packet from some other node *i* contain *i’s* current shortest path costs to all other network nodes.

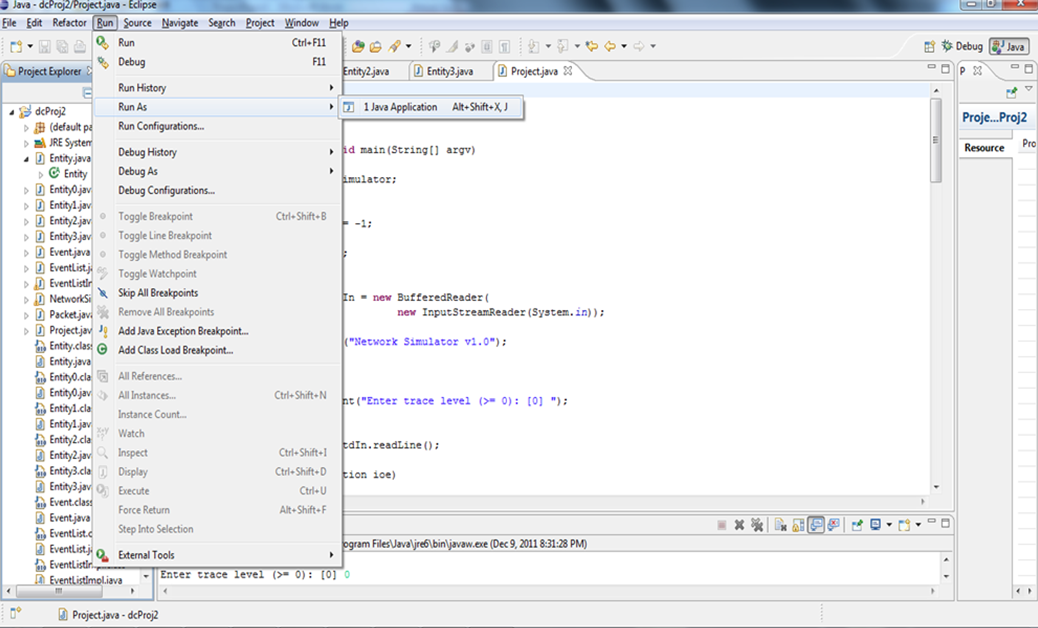
Update() method uses these received values to update its own distance table. If its own minimum cost to another node changes as a result of the update, this node informs its directly connected neighbors of its change in minimum cost by sending them a routing packet.

***getMinCost* ()**

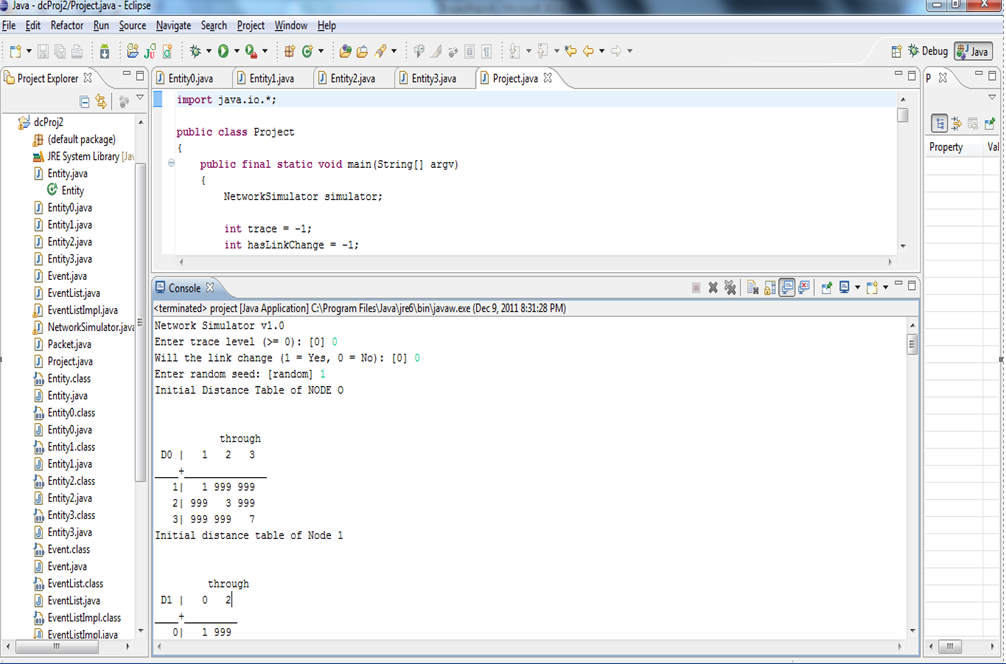
This method, which finds the min cost to the other nodes from this node using the distance table.

**Running Conditions:**

Run the project.java file by selecting the menu –Run and submenu Run configurations as shown below



**Sample Output:**



Network Simulator v1.0

Enter trace level (>= 0): [0] 0

Will the link change (1 = Yes, 0 = No): [0] 0

Enter random seed: [random] 1

Initial Distance Table of NODE O

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 999 999

2| 999 3 999

3| 999 999 7

Initial distance table of Node 1

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 999

2| 999 1

3| 999 999

Initial distinace Table of Node 2

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 999 999

1| 999 1 999

3| 999 999 2

Initial distance tavle of Node 3

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 999

1| 999 999

2| 999 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 999

2| 999 3 999

3| 999 5 7

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 999

1| 8 999

2| 10 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 999

2| 2 3 999

3| 999 5 7

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 999 999

1| 4 1 999

3| 10 999 2

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 999

1| 8 999

2| 10 2

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 999

2| 4 1

3| 8 999

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 999 999

1| 4 1 999

3| 8 999 2

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 999

1| 8 999

2| 9 2

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 999 9

1| 4 1 999

3| 8 999 2

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 5

1| 8 3

2| 9 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 999

2| 2 3 9

3| 999 5 7

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 4

2| 4 1

3| 8 3

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 9

1| 4 1 999

3| 8 999 2

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 9

1| 4 1 10

3| 8 999 2

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 9

1| 4 1 10

3| 8 9 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 999

2| 2 3 9

3| 9 5 7

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 3

2| 4 1

3| 8 3

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 3

2| 4 1

3| 6 3

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 999

2| 2 3 9

3| 4 5 7

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 4

1| 8 3

2| 9 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 15

2| 2 3 9

3| 4 5 7

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 9

1| 4 1 10

3| 8 9 2

through

D3 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_

0| 7 4

1| 8 3

2| 9 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 15

2| 2 3 9

3| 4 5 7

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 9

1| 4 1 10

3| 8 4 2

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 7

1| 4 1 5

3| 8 4 2

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 3

2| 3 1

3| 6 3

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 7

1| 4 1 5

3| 7 4 2

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 10

2| 2 3 9

3| 4 5 7

through

D2 | 0 1 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_

0| 3 2 6

1| 4 1 5

3| 7 4 2

through

D1 | 0 2

\_\_\_\_+\_\_\_\_\_\_\_\_\_

0| 1 3

2| 3 1

3| 5 3

through

D0 | 1 2 3

\_\_\_\_+\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1| 1 4 10

2| 2 3 9

3| 4 5 7

Simulator terminated at t=28.94791301701228, no packets in medium.

**Problems:**

*Counting to infinity*: This condition would continuously loop for the packets around the network, even when the destination network is down. While the routers are counting to infinity, the invalid information allows a routing loop to exist.

*Routing Loops*: Routing loops can occur if the internet work is slow convergence on a new configuration causes inconsistence routing entry.