Jessica Daniels

CSCI 4431

Project 2

**Implementing A Routing Algorithm**

**Design Document**

The distance vector routing algorithm is executed in an emulated environment. Source files prog3.c, node0.c, node1.c, node2.c, and node3.c are needed for the routing algorithm to be executed.

Prog3.c represents an emulated network environment. It produces the network layer 2, and the transmission and delivery between connected nodes. At the start of the emulated environment it calls on functions rtinit0(), rtinit1(), rtinit2(), and rtinit3().

The emulated environment calls the rtinit functions to connected to the 4 nodes, which is represented by the four source files, node0.c, node1.c, node2.c, and node3.c. Each of the nodes contain the following functions:

***rtinit()***

This function is called once at the start of the emulated environment to initialize the data structure of the node. It initializes the cost of other connected nodes to tell the node what are the other nodes it is physically connected to. It determines when a node is connected if it has a specified cost and knows when a node is not connected when the cost is set to infinity.

***struct distance\_table dt0***

The cost of the nodes is recorded in the *distance\_table* data structure. The table is designed for the rows to represent the destinations in the network and the columns is built for the directly connected nodes.

The node has all the cost to connect to the nodes in the network but least cost, or the shortest path, to every nodes in the row is represented in the *mincost[]* array. When rtinit() initializes the cost to nodes, it sends a packet containing the *mincost[]* and its values to their directly connected nodes. Therefore each node sends and receives the shortest paths values for each node it is connected to at clocktime=.

The initial *distance\_table* and *mincost[]* for each of the nodes initially produces the following output:

Node 0:

via

D0 |    1     2    3

  ----|---------------------

     1|    1   9999   9999

dest 2|  9999     3   9999

     3|  9999   9999     7

Node 1:

via

   D1 |    0     2

  ----|-----------------

     0|    1   9999

dest 2|  9999     1

     3|  9999   9999

Node 2:

                via

   D2 |    0     1    3

  ----|---------------------

     0|    3   9999   9999

dest 1|  9999     1   9999

     3|  9999   9999     2

Node 3:

                via

   D3 |    0     2

  ----|-----------------

     0|    7   9999

dest 1|  9999   9999

     2|  9999     2

***rtupdate()***

Each of the nodes initially has the following values as the shortest path in the *mincost[]* array:

Node 0:

mincost[0, 1, 3, 7]

Node 1:

mincost[1, 0, 1, 9999]

Node 2:

mincost[3, 1, 0, 2]

Node 3:

mincost[7, 9999, 2, 0]

These values are the values in the packet the nodes send to their connected neighbor nodes. Once the node receives a packet from another node, it executes the rtupdate() function to update the values in the distance table.

***rtupdate()***

When a node receive a packet with the values of nodes shortest path, it compares the values it received to the values in the have located in *distance\_table.* Each node uses the following equation to calculate the distance table:

dt0.costs[i][source] = pkt2send0.mincost[source] + rcvdpkt->mincost[i];

where *source* is the node that sent the packet. So if node 0 receive a packet from node 1, which would be mincost[1, 0, 1, 9999], rtupdate0() would compare the values it has in the distance table using the following equation as such:

dt0.costs[0][1] = 1 + 1 = 2;

dt0.costs[1][1] = 1 + 0 = 1;

dt0.costs[2][1] = 1 + 1 = 2;

dt0.costs[3][1] = 1 + 9999 = 10000;

so in the *distance\_table* for node 0, the column for node 1 gets updated to:

via

D0 | 1 2 3

-----|-----------------------------

1 | 1 9999 9999

dest 2 | 2 3 9999

3 | 10000 9999 7

and the new *mincost[]* for node 0 would be:

mincost[0, 1, 2, 7]

because the values in *mincost[]* changed, it resends a packet to the connected nodes. Each of the packets continually resend and update to get the final distance\_table for the nodes:

Node 0:

via

D0 |    1     2    3

  ----|---------------------

     1|    1   4   10

dest 2|  2     3   9

     3|  4   5      7

Node 1:

via

   D1 |    0     2

  ----|-----------------

     0|    1   3

dest 2|  3      1

     3|  5   3

Node 2:

                via

   D2 |    0     1    3

  ----|---------------------

     0|    3   2   6

dest 1|  4     1   5

     3|  7   4     2

Node 3:

                via

   D3 |    0     2

  ----|-----------------

     0|    7   4

dest 1|  8   3

     2|  9     2