Lab3: Guideline to Implement Distance Vector Routing Algorithm

By Kyoomars Noghani Alizadeh and Dejene Boru Oljira

The purpose of this document is to clarify your task in the third lab (Lab 3). All these information are extracted from the original lab descriptions.

Initialization

Each node has 2 main functions that you have to implement: 1) rtinit() and 2) rtupdate().

rtinit() is the initialization function that will be called once at the beginning of the simulation. As the function name implies it should initialize the distance table of a node to reflect its **direct distance** to its neighbors.

Note: Distance table of node 0 is "dt0" that you can find its definition at the beginning of the code where entry "`dt0.costs[i][j]`" is node 0's currently computed cost to node "*i*" via **direct neighbor** *"j"*.

Hint: For example, "**`dt0`**`.costs[2][2]=3`" which says that the **node0** can reach to node 2 (i index) via node 2 (j index) with 3 units.

Hint: If a node does not have a direct link to its neighbors, for instance consider node 1 and 3 in Fig.1, then the initial distance between these nodes is INFINITE

Sending the first message

Once the routing array inside the node is initialized, the router calculates the minimum value to reach **each neighbor** and sends the minimum distance value to its neighbors.

This happens through preparing messages of type *"struct rtpkt"* and calling *"tolayer2(rtpkt)"* function.

Note: You call *"tolayer2"* function since a router has physical, link and network layer and you are working in the network layer.

Note 2: You can find *"tolayer2"* function in prog3.c file.

Updating neighbors

rtupdate(): The message you sent in the previous step will be received by the neighbor routers through the *"rtupdate()"* function. Upon receiving the message the router parses the information inside the packet, updates its local distance array, and redistributes the updated information **if and only if** the local distance array is updated with new values.

Printing

Function “printdt()” helps you to trace the iteration of the updates and debug your code.

Expected Output:

For your sample output, your procedures should print out a message whenever your rtinit() or rtupdate() procedures are called, giving the time (define a global variable clocktime).

For rtupdate() functions you should print the identity of the sender of the routing packet that is being passed to your routine, whether or not the distance table is updated, the contents of the distance table, and a description of any messages sent to neighboring nodes as a result of any distance table updates.

General Notes

Note: A tracing value of 2 may be helpful to you in debugging your code.

Note 2: Your program will run until there are no more routing packets in-transit in the network, at which point the emulator will terminate.

Note 3: There is no packet loss and corruption in this assignment. So, you will receive the packets in the correct order safe and sound.

Starter code for Node0 (Note: You are not obliged to use this start code, you are free to implement your own!)

```
#include < stdio.h >
  extern struct rtpkt {
    int sourceid; /* sender router id */
    int destid; /* id of neighbor router (must be an immediate neighbor) */
    int mincost[4]; /* min cost to node 0 ... 3 */
  };
extern int TRACE;
extern int YES;
extern int NO;
extern float clocktime;
struct distance_table {
  /*costs[i][j] := cost to go to node i via direct neighbour j*/
  int costs[4][4];
}dt0;
/*utility min funciton*/
int min(int a, int b, int c) {
  int temp = (a <= b ? a : b);
  temp = (temp <= c ? temp : c);
  return temp;
}
/* students to write the following two routines, and maybe some others */
void rtinit0() {
  printf("\nrtinit0 invoked at time %f \n", clocktime);
  /*initialize distance table*/
  for (int i = 0; i < 4; i++)
    for (int j = 0; j < 4; j++)
      dt0.costs[i][j] = /*fill here*/;
  /*updating costs based on given initial network topology*/
  /*Complete here*/
  /*printing the initial distance table*/
  printdt0( & dt0);
  /*make dv packets to send to neighbours*/
  struct rtpkt pkt1, pkt2, pkt3;
  /*set source and dest for each pkt*/
  /*Complete here*/
  /*set minimum cost for each */
  for (int i = 0; i < 4; i++) {
    /*complete here*/
  }
  /*sending dv packets to neighbours (call tolayer2)*/

  /*Complete here*/
}
void rtupdate0(rcvdpkt)
{
  printf("\nrtpdate0 invoked at time %f \n", clocktime);
  //wrong destination check
  /*complete this*/
  return;
  /*flag is set to 1 in case there is an update in the distance table*/
  int flag = 0;
  printf("\nPacket rcvd at node 0 from node %d \n", rcvdpkt - > sourceid);
  /*Traversing column j of distance table to see if it needs to be updated*/
  for (int i = 0; i < 4; i++) {
    /* complete here*/
  }
  //change in dt0 detected!
  if (flag == 1) {
    printf("\nDistance table at node 0 updated: \n");
    printdt0( & dt0);
    struct rtpkt pkt1, pkt2, pkt3;
    /*set src and dest*/
    /*fill here*/
    /*set min cost to itself (node0)*/
    /*fill here*/
    /*compute minimum cost to neighbors*/
    for (int i = 1; i < 4; i++) {
      /*complete here*/
    }
    /*send dv packets to neighbors*/
    /*complete here*/
  }
}
```