

node1.c

```
#include <stdio.h>

extern struct rtpkt {
    int sourceid;          /* id of sending router sending this pkt */
    int destid;            /* id of router to which pkt being sent
                           (must be an immediate neighbor) */
    int mincost[4];        /* min cost to node 0 ... 3 */
};

extern int TRACE;
extern int YES;
extern int NO;

static struct distance_table
{
    int costs[4][4];
} dt1, *distance;

/* students to write the following two routines, and maybe some others */
// define 999 as infinity
#define INF 999

//extranal function and variable
extern void tolayer2(struct rtpkt packet);
extern float clocktime;

//local functions and variables
void rtinit1();
void rtupdate1(struct rtpkt *rcvpkt);
void printdt1(struct distance_table *dtptr);
void updata1();
void printf_sendinfo1();
int compute_shortest_path1();

static int link[4];        //cost to neighbor
static int shortest[4];    //smallest cost to destination

/*
 * initialize router table
 */
void rtinit1()
{
    int destination;        // destination node id 0, 1, 2, 3
    int neighbor;           // neighbor node id 0, 1, 2, 3

    //print the time function was called
    printf("time=%f> rtinit1\n", clocktime);

    //initialize dt
    distance = &dt1;

    // initialize the link costs to neighbor
```

```
link[0] = 1;
link[1] = 0;
link[2] = 1;
link[3] = INF;

//initialize the router table
for (destination = 0; destination < 4; destination++)
{
    for (neighbor = 0; neighbor < 4; neighbor++)
    { //if destination is neighbor, cost is link cost, else is infinity
        if (destination == neighbor) {
            distance->costs[destination][neighbor] = link[destination];
        }
        else {
            distance->costs[destination][neighbor] = INF;
        }
    }

    //at initialization step, the shortest path is link cost
    shortest[destination] = link[destination];
}

//update router table
update1();
//print router table
printdt1(distance);
}

void rtupdate1(struct rtpkt *rcvpkt)
{
    int idx;
    //get source id
    int src = rcvpkt->sourceid;;

    // print the time function was called
    printf("time=%f> rtupdate1: node1 receiving a packet from node%d\n", clocktime, src);

    //update router table
    for (idx = 0; idx < 4; idx++)
    {
        distance->costs[idx][src] = link[src] + rcvpkt->mincost[idx];
        if (distance->costs[idx][src] > INF)
            distance->costs[idx][src] = INF;
    }

    // print router table
    printdt1(distance);

    //if shortest path changed, update router table
    if (compute_shortest_path1())
        update1();
}

/* compute the shortest path */
```

```

int compute_shortest_path1()
{
    int destination;        // destination node id 0, 1, 2, 3
    int neighbor;           // neighbor node id 0, 1, 2, 3
    int lowestCost;         // save shortest path and compare with recorder
    int update = 0;         // whether shortest path changed; 0 is not changed; 1 is changed

    for (destination = 0; destination < 4; destination++)
    {
        lowestCost = distance->costs[destination][0];

        for (neighbor = 1; neighbor < 4; neighbor++)
        {
            if (lowestCost > distance->costs[destination][neighbor])
                lowestCost = distance->costs[destination][neighbor];
        }

        //if shortest patch changed, update shortest path
        if (lowestCost != shortest[destination]) {
            shortest[destination] = lowestCost;
            //mark shortest path changed
            update = 1;
        }
    }

    return update;
}

/*
 * update the packet and send it to the other nodes
 */
void updata1()
{
    int node;               // node id in the network 0, 1, 2, 3
    struct rtpkt pkt, *p;   //packet

                                //set packet source is node 1
    p = &pkt;
    p->sourceid = 1;

    //set mincost information in packet
    for (node = 0; node < 4; node++)
    {
        p->mincost[node] = shortest[node];
    }

    //send packet to node 0
    p->destid = 0;
    tolayer2(*p);
    printf_sendinfo1(p);
    //send packet to node 2
    p->destid = 2;
    tolayer2(*p);
    printf_sendinfo1(p);
}

```

```

/*
 * print send information "%time %source send packet to %destination with cost %mincost"
 */
void printf_sendinfo1(struct rtpkt *p)
{
    printf("time=%f> node%d sent packet to node%d with following minimum costs: %d\n",
        clocktime, p->sourceid, p->destid, p->mincost[p->destid]);
}

void printdt1(dtptr)
struct distance_table *dtptr;

{
    printf("          via  \n");
    printf(" D1 |    0    2 \n");
    printf(" ----|-----\n");
    printf("    0|  %3d  %3d\n", dtptr->costs[0][0], dtptr->costs[0][2]);
    printf("dest 2|  %3d  %3d\n", dtptr->costs[2][0], dtptr->costs[2][2]);
    printf("    3|  %3d  %3d\n", dtptr->costs[3][0], dtptr->costs[3][2]);
}

linkhandler1(linkid, newcost)
int linkid, newcost;
/* called when cost from 1 to linkid changes from current value to newcost*/
/* You can leave this routine empty if you're an undergrad. If you want */
/* to use this routine, you'll need to change the value of the LINKCHANGE */
/* constant definition in prog3.c from 0 to 1 */

{
    int node;
    int previousCost;

    /* print the time function was called */
    printf("time=%f> linkhandler1\n", clocktime);

    /* get the new cost */
    previousCost = link[linkid]; /* save the previous cost */
    link[linkid] = newcost;      /* replace it with new cost */

                                /* update the distance table */
    for (node = 0; node<4; node++)
    {
        distance->costs[node][linkid] = distance->costs[node][linkid] - previousCost + newcost;
        if (distance->costs[node][linkid] > INF)
            distance->costs[node][linkid] = INF;
    }

    /* print the information */
    printdt1(distance);                                /* print the distance table */
                                                        /* compute the shortest path */
    if (compute_shortest_path1())                        /* if shortest path found */
        updata1();                                       /* build the packet and send it to the other nodes */
}

```