

# Implementing a Routing Protocol

## Project Object

이번 프로젝트는 교재 p412에 나와있는 'The Distance-Vector(DV) Routing Algorithm'을 구현하는 것이다. 이 알고리즘에 대한 규칙은 다음과 같다.

1. For each neighbor  $v$ , the cost  $c(x,v)$  from  $x$  to directly attached neighbor,  $v$
2. Node  $x$ 's distance vector, that is,  $D_x = [D_x(y): y \in N]$  containing  $x$ 's estimate of its cost to all destinations,  $y$ , in  $N$
3. The distance vectors of each of its neighbors, that is,  $D_v = [D_v(y): y \in N]$  for each neighbor  $v$  of  $x$

```
Distance-Vector (DV) Algorithm
At each node, x:
1 Initialization:
2   for all destinations y in N:
3     Dx(y) = c(x,y) /* if y is not a neighbor then c(x,y) = ∞ */
4   for each neighbor w
5     Dw(y) = ? for all destinations y in N
6   for each neighbor w
7     send distance vector Dx = [Dx(y): y in N] to w
8
9 loop
10  wait (until I see a link cost change to some neighbor w or
11      until I receive a distance vector from some neighbor w)
12
13  for each y in N:
14    Dx(y) = minv{c(x,v) + Dv(y)}
15
16 if Dx(y) changed for any destination y
17   send distance vector Dx = [Dx(y): y in N] to all neighbors
18
19 forever
```

교재 p413에는 이 알고리즘에 대한 pseudo 코드가 위 사진과 같이 나와있다. 위 코드의 순서는 다음과 같다.

1. Initialization: 직접 연결되어있는 neighbor node에게 cost 정보를 받아 distance table을 세팅한다.
2. Distance table을 완성시킨 후 neighbor node에게 다시 전송한다.
3. neighbor node에게 다시 cost 정보를 받아 distance table을 계산 후 다시 전송한다.
4. 2, 3 번 과정을 계속 반복한다.

## Source Code

### 1. node0.c (printdt0() 함수 제외)

- node1.c, node2.c, node3.c 는 아래와 변수 명만 다르고 구조 같아서 하드 카피로 제출

```
#include <stdio.h>
#include "project3.h"

extern int TraceLevel;
extern float clocktime;

struct distance_table {
    int costs [MAX_NODES] [MAX_NODES];
};

struct distance_table dt0;
struct NeighborCosts *neighbor0;

/* students to write the following two routines, and maybe some others */

struct RoutePacket packet; //set packet as local variable

void printdt0( int MyNodeNumber, struct NeighborCosts *neighbor, struct
distance_table *dtptr );

void rtinit0() {
    int i, j;
    struct distance_table *thisnode;

    thisnode = &dt0; //set struct pointer
    printf("*****\n");
    printf("At time t = %f, rtinit0() called\n", clocktime);

    // initiate minimum cost and distance table as INFINITY
    for (i = 0; i < 4; i++) {
        packet.mincost[i] = INFINITY;
        for (j = 0; j < 4; j++) {
            thisnode->costs[i][j]= INFINITY;
        }
    }

    neighbor0 = getNeighborCosts(0); //get neighborcosts from project3.c

    // set neighborcosts in distance table
    for (i = 0; i < 4; i++) {
        thisnode->costs[i][i] = neighbor0->NodeCosts[i];
    }

    // check each distance from node i to node j in comparison to the packet
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
```

```

        if (packet.mincost[i] > thisnode->costs[i][j]) {
            packet.mincost[i] = thisnode->costs[i][j];
        }
    }

packet.sourceid = 0; //set packet's sourceID
for (i = 1; i < MAX_NODES; i++) { //start from 1 bc this node# is 0
    packet.destid = i;
    toLayer2(packet); //use tolayer2 to send to another node
    printf("At time t = %f, node %d sends packet to node %d
with: %d %d %d\n", clocktime, packet.sourceid, i , packet.mincost[0],
packet.mincost[1], packet.mincost[2], packet.mincost[3]);
}
printf(dt0(0, neighbor0, thisnode));
printf("*****\n");
printf("\n\n");
}

void rtupdate0( struct RoutePacket *rcvdpkt ) {
    int i, j, src, update = 0;
    struct distance_table *thisnode;

    thisnode = &dt0;

    for (i = 0; i < 4; i++) {
        packet.mincost[i] = INFINITY;
    }
    printf("*****\n");
    printf("At time t = %f, rtupdate0() called\n", clocktime);

    // check received source ID
    src = rcvdpkt->sourceid;
    printf("Received packet from node %d\n", src);

    for (i = 0; i < 4; i++) {
        // if there is new mincost change, update distance table.
        if (thisnode->costs[i][src] > (thisnode->costs[src][src] + rcvdpkt-
>mincost[i])) {
            thisnode->costs[i][src] = (thisnode->costs[src][src] + rcvdpkt-
>mincost[i]);
            update = 1;
        }
    }

    // If we update distance table above, change this node's packet mincost
    if (update == 1) {
        for (i = 0; i < 4; i++) {
            for (j = 0; j < 4; j++) {

```

```

        if (packet.mincost[i] > thisnode->costs[i][j]) {
            packet.mincost[i] = thisnode->costs[i][j];
        }
    }

    packet.sourceid = 0; //set packet's sourceID
    for (i = 1; i < MAX_NODES; i++) { //start from 1 bc this node# is 0
        packet.destid = i;
        toLayer2(packet); //use toLayer2 to send to another node
        printf("At time t = %f, node %d sends packet to node %d
with: %d %d %d\n", clocktime, packet.sourceid, i , packet.mincost[0],
packet.mincost[1], packet.mincost[2], packet.mincost[3]);
    }
}

printfdt0(0, neighbor0, thisnode);
printf("*****\n");
printf("\n\n");
}

```

## Result screen

결과 분석에 더 많은 result screen 이 있습니다.

```

sungminryu:~/project$ ./project3 1
*****
At time t = 0.000000, rtinit0() called
At time t = 0.000000, node 0 sends packet to node 1 with: 0 1 3 7
At time t = 0.000000, node 0 sends packet to node 2 with: 0 1 3 7
At time t = 0.000000, node 0 sends packet to node 3 with: 0 1 3 7
    via
    00 |   1   2   3
    ---|-----
dest 1|   1   9999  9999
dest 2|  9999   3   9999
dest 3|  9999  9999   7
*****


***** time t = 0.000000, rtinit1() called
At time t = 0.000000, node 1 sends packet to node 0 with: 1 0 1 9999
At time t = 0.000000, node 1 sends packet to node 2 with: 1 0 1 9999
    via
    01 |   0   2
    ---|-----
dest 0|   1   9999
dest 2|  9999   1
dest 3|  9999  9999
*****


***** time t = 0.000000, rtinit2() called
At time t = 0.000000, node 2 sends packet to node 0 with: 3 0 0 2
At time t = 0.000000, node 2 sends packet to node 1 with: 3 0 0 2
At time t = 0.000000, node 2 sends packet to node 3 with: 3 1 0 2
    via
    02 |   0   1   3
    ---|-----
dest 0|   3   9999  9999
dest 1|  9999   1   9999
dest 3|  9999  9999   2
*****
```

[./project3 1 로 실행한 모습(처음 부분)]

```

1.bash

*****
***** At time t = 24.449234, rtupdate0() called
Received packet from node 2
    via
    00 |   1   2   3
    ---|-----
dest 1|   1   4   10
dest 2|   2   3   9
dest 3|   4   5   7
*****
***** At time t = 24.984856, rtupdate0() called
Received packet from node 1
    via
    00 |   1   2   3
    ---|-----
dest 1|   1   4   10
dest 2|   2   3   9
dest 3|   4   5   7
*****
***** At time t = 25.775383, rtupdate0() called
Received packet from node 3
    via
    00 |   1   2   3
    ---|-----
dest 1|   1   4   10
dest 2|   2   3   9
dest 3|   4   5   7
*****
Simulator terminated at t=25.775383, no packets in medium

```

[./project3 1 로 실행한 모습(마지막 부분)]

./project3 1 > output.txt 명령어로 위 실행 결과를 텍스트 파일로 첨부해서 하드 카피로 제출했습니다.

## Results analysis and discussion

이번 프로젝트에서는 4개의 노드의 node#.c 파일을 채워 넣으면 된다. 각각의 c 파일은 rtinit#() 과 rtupdate#()함수로 이루어져 있다. 이 함수들은 내용이 모두 동일하기 때문에 node0.c에 대해 설명하겠습니다.

### 1. rtinit0() 함수

rtinit0() 함수는 node0.c 파일에 있는 함수로 초기 node0 의 distance table 을 초기화하고 이 데이터를 전송하는 과정을 수행한다.

```

void rtinit0() {
    int i, j;
    struct distance_table *thisnode;

    thisnode = &dt0; //set struct pointer
    printf("*****\n");
    printf("At time t = %f, rtinit0() called\n", clocktime);

    // initiate minimum cost and distance table as INFINITY
    for (i = 0; i < 4; i++) {

```

```

packet.mincost[i] = INFINITY;
for (j = 0; j < 4; j++) {
    thisnode->costs[i][j] = INFINITY;
}
}

neighbor0 = getNeighborCosts(0); //get neighborcosts from project3.c

// set neighborcosts in distance table
for (i = 0; i < 4; i++) {
    thisnode->costs[i][i] = neighbor0->NodeCosts[i];
}

// check each distance from node i to node j in comparison to the packet
for (i = 0; i < 4; i++) {
    for (j = 0; j < 4; j++) {
        if (packet.mincost[i] > thisnode->costs[i][j]) {
            packet.mincost[i] = thisnode->costs[i][j];
        }
    }
}

packet.sourceid = 0; //set packet's sourceID
for (i = 1; i < MAX_NODES; i++) { //start from 1 bc this node# is 0
    packet.destid = i;
    toLayer2(packet); //use tolayer2 to send to another node
    printf("At time t = %f, node %d sends packet to node %d
with: %d %d %d %d\n", clocktime, packet.sourceid, i, packet.mincost[0],
packet.mincost[1], packet.mincost[2], packet.mincost[3]);
}
printf("\n\n");
}

```

위 함수에 대한 과정은 다음과 같다.

1. 초기 minimum cost 와 distance table 을 모두 INFINITY(9999)로 세팅한다.
  2. getNeighborCosts() 함수를 이용해서 neighbor 에 대한 정보를 neighbor0 가 가리키는 pointer structure 에 저장한다.
  3. 이 정보를 이용해 distance table 을 업데이트한다.
  4. 업데이트한 정보를 tolayer2() 함수를 이용해서 주변 노드로 보내준다.
- tolayer2() 함수를 이용할 때, 자신에게 보내거나 연결되어 있지 않은 노드로 보내면 오류가 나므로 이 부분의 코드가 약간 다르다. 예를 들면, 1 번과 3 번 노드들은 연결되어 있지 않아 각각으로 데이터를 보내는 과정을 생략했다.

```

*****
At time t = 0.000000, rtinit0() called
At time t = 0.000000, node 0 sends packet to node 1 with: 0 1 3 7
At time t = 0.000000, node 0 sends packet to node 2 with: 0 1 3 7
At time t = 0.000000, node 0 sends packet to node 3 with: 0 1 3 7
    via
    D0 |   1   2   3
    ----|-----
dest 1|   1 9999 9999
dest 2| 9999   3 9999
dest 3| 9999 9999   7
*****

```

```

*****
At time t = 0.000000, rtinit1() called
At time t = 0.000000, node 1 sends packet to node 0 with: 1 0 1 9999
At time t = 0.000000, node 1 sends packet to node 2 with: 1 0 1 9999
    via
    D1 |   0   2
    ----|-----
dest 0|   1 9999
dest 2| 9999   1
dest 3| 9999 9999
*****

```

```

*****
At time t = 0.000000, rtinit2() called
At time t = 0.000000, node 2 sends packet to node 0 with: 3 1 0 2
At time t = 0.000000, node 2 sends packet to node 1 with: 3 1 0 2
At time t = 0.000000, node 2 sends packet to node 3 with: 3 1 0 2
    via
    D2 |   0   1   3
    ----|-----
dest 0|   3 9999 9999
dest 1| 9999   1 9999
dest 3| 9999 9999   2
*****

```

```

*****
At time t = 0.000000, rtinit3() called
At time t = 0.000000, node 3 sends packet to node 0 with: 7 9999 2 0
At time t = 0.000000, node 3 sends packet to node 2 with: 7 9999 2 0
    via
    D3 |   0   2
    ----|-----
dest 0|   7 9999
dest 1| 9999 9999
dest 2| 9999   2
*****

```

rtinit#()은 초기에 한 번만 수행하기 때문에 총 4 개의 결과(node 가 4 개 이므로)를 위와 같이 첨부했다. 자세히 보면 각 노드들이 주변 정보를 받아 초기화를 시켜준 것을 볼 수 있고 초기화 후 직접 연결 되어 있는 노드들로 데이터를 보낸 것(노란색 박스)을 볼 수 있다.

## 2. rtupdate0() 함수

rtupdate0() 함수는 node0.c 파일에 있는 함수로 주변 노드들의 업데이트 된 데이터를 받아와서 distance table 을 재구성하고 이 데이터를 다시 전송하는 과정을 수행한다.

```
void rtupdate0( struct RoutePacket *rcvdpkt ) {
    int i, j, src, update = 0;
    struct distance_table *thisnode;
    thisnode = &dt0;

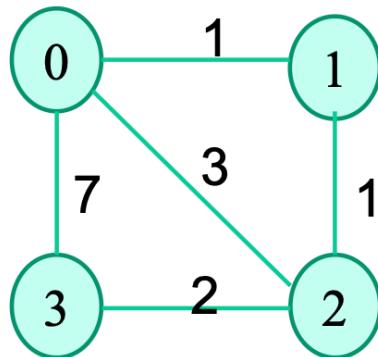
    for (i = 0; i < 4; i++) {
        packet.mincost[i] = INFINITY;
    }
    printf("*****\n");
    printf("At time t = %f, rtupdate0() called\n", clocktime);
    // check received source ID
    src = rcvdpkt->sourceid;
    printf("Received packet from node %d\n", src);

    for (i = 0; i < 4; i++) {
        // if there is new mincost change, update distance table.
        if (thisnode->costs[i][src] > (thisnode->costs[src][src] + rcvdpkt-
>mincost[i])) {
            thisnode->costs[i][src] = (thisnode->costs[src][src] + rcvdpkt-
>mincost[i]);
            update = 1;
        }
    }
    // If we update distance table above, change this node's packet mincost
    if (update == 1) {
        for (i = 0; i < 4; i++) {
            for (j = 0; j < 4; j++) {
                if (packet.mincost[i] > thisnode->costs[i][j]) {
                    packet.mincost[i] = thisnode->costs[i][j];
                }
            }
        }
        packet.sourceid = 0; //set packet's sourceID
        for (i = 1; i < MAX_NODES; i++) { //start from 1 bc this node# is 0
            packet.destid = i;
            toLayer2(packet); //use tolayer2 to send to another node
            printf("At time t = %f, node %d sends packet to node %d
with: %d %d %d %d\n", clocktime, packet.sourceid, i , packet.mincost[0],
packet.mincost[1], packet.mincost[2], packet.mincost[3]);
        }
    }
    printdt0(0, neighbor0, thisnode);
    printf("*****\n");
    printf("\n\n");
}
```

`rtupdate0()` 함수는 `rtinit0()` 함수와 다르게 input 값이 있다. 이 input은 다른 노드에서 보낸 데이터 패킷으로 보낸 노드의 정보와 cost 정보가 들어있다. 따라서 아래와 같이 보내온 노드가 어떤 노드인지를 알아낼 수 있다.

```
src = rcvdpkt->sourceid;
```

만약 받은 패킷에 정보가 distance table에 영향을 준다면 update를 해야하는데, 책에 나와 있는  $Dx(y) = \min\{ c(x, y) + Dv(y) \}$ 를 이용해서 update를 해준다. 만약 update가 되었다면 업데이트된 데이터 packet을 연결되어 있는 노드에게 전송해야 하므로 다시 `toLayer2()` 함수를 사용해 `rtinit0()`에서 보낸 방법과 같은 방법으로 전송한다.



노드 구성은 위와 같이 되어있으므로 결과는 아래와 같이 나왔다.

```
*****
At time t = 25.775383, rtupdate0() called
Received packet from node 3
      via
      D0 |   1   2   3
      ---|-----
dest 1|   1   4   10
dest 2|   2   3   9
dest 3|   4   5   7
*****
Simulator terminated at t=25.775383, no packets in medium
sungminryu::project$
```

위 테이블은 node0에서 1, 2, 3 노드를 통해(via) 목적지(dest #)에 도달하는 최단 경로를 나타낸다. 직접 계산한 결과와 코드로 시뮬레이션 한 결과가 일치하게 나온 것을 알 수 있다. 나머지 결과는 output.txt 파일로 제출했습니다.

## Reference

[1] Kurose & Ross, Computer Networking: A Top-Down Approach, 7th edition