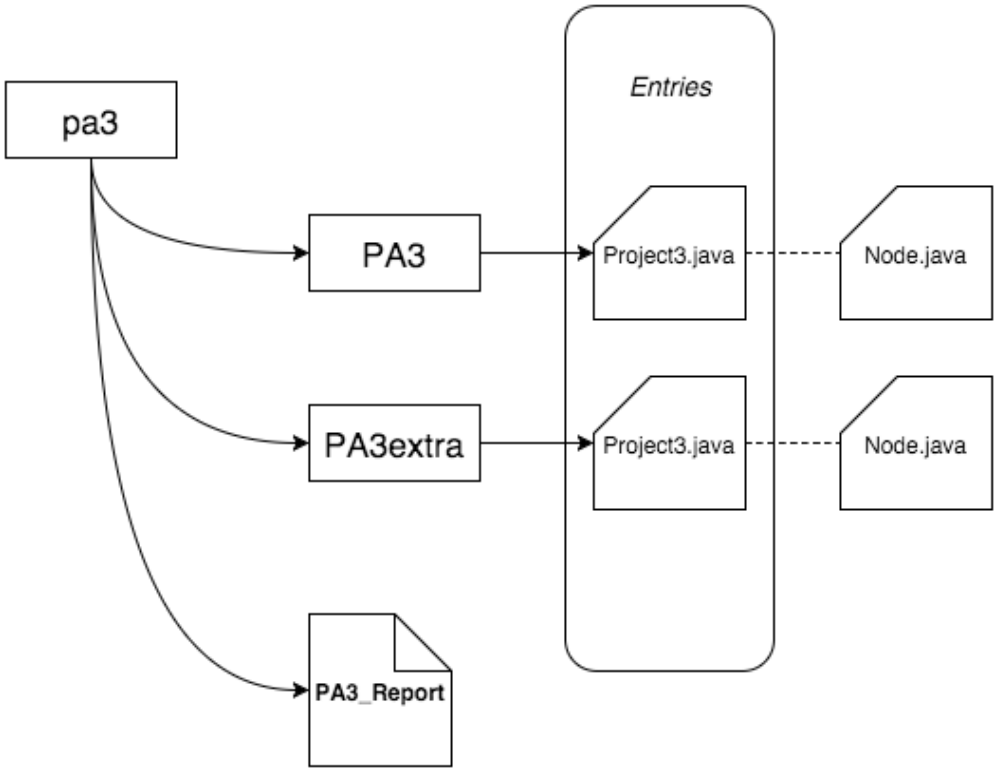
PA3 Report

Before all:

This report contains two parts to explain and report our programs, results and logistics:

- Part1 CS455 Basic Routing (Distance-Vector)
- Part2 CS655 Split Horizon with Poison Reverse (Pre-final Node)

Our gsubmit accounts are "yt80" and "rexwang". Each part above has instructions to run our program, our output results (final distance table), logistics. And our structure of files are as follows, and among the files we **only modified** "Node.java" files to finish each approach, which can be compiled to: "Node.class".



Compile Instructions for all:

According to our file structure showed above, we created separate folders for each routing. Set Basic DV Routing ("PA3" folder) as example, since "main ()" function is included in "Project3.java", to compile, use "javac *.java" to compile all the java files. To start, use" java Project3". The compilation process of the other routing program is very similar except "cd" in different directories.

(Note that our default pseudorandom seed is 3322)

〔Out outputs are directly printed to the screen, so we don't need to clean any ".txt" files〕

```
➜ pa3  ls
PA3      PA3extra
➜ pa3  cd PA3
➜ PA3  ls
Event.java            NetworkSimulator.java Packet.java
EventList.java        Node.java             Project3.java
EventListImpl.java    OSIRandom.java        min_parameter.java
➜ PA3  javac Project3.java
➜ PA3  java Project3
Network Simulator v1.0
Enter trace level (>= 0): [0] 2
Enter seed (>= 0): [3322]
```

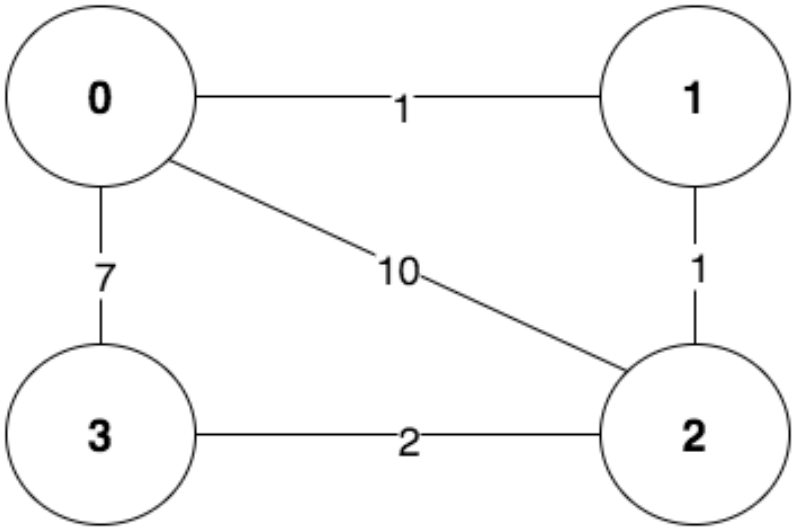
```
➜ pa3  ls
PA3      PA3extra
➜ pa3  cd PA3extra
➜ PA3extra  ls
Event.java            EventListImpl.java    Node.java             Packet.java
 min_parameter.java    table_ds.java
EventList.java        NetworkSimulator.java OSIRandom.java        Project3.java
 node_com.java         test.java
➜ PA3extra  javac Project3.java
➜ PA3extra  java Project3
Network Simulator v1.0
Enter trace level (>= 0): [0] 3
Enter seed (>= 0): [3322]
```

Part1. Distance-Vector Routing

Working correctness:

Each node initials correctly, and from node 0 to node 3, they all converge correctly with link changes and without link changes. Besides, poison reverse works correctly too (note we use integer value 9999 to represent 'infinity', each next hop of any node has distance of 9999 through that node).

Overall Designation:



- The functions and variables:

void rtinit(int nodename, int[] initial_lkcost)

This function is to initialize the distance table, the arguments are nodename (from 0 to 3), and their link costs. Then, it sends nodes' directly-connected neighbor nodes the minimum-cost to all other net work nodes. Here, this function calls NetworkSimulator.tolayer2 to send the information in a routing packet. After the whole initialization, it prints (specific node initialization) information.

void rtupdate(Packet rcvdpkt)

This function will be called when the nodes receive in a routing packet that was sent to it by one of its directly connected neighbors. The function's

parameter "rcvdpkt" is a pointer to the packet that was received. This function use received values to update specific "current" node's own distance table. And this function calls "Updateminsoct()" to judge whether to update table. If it's not need to update, we mark the node with "********silence" tag.

protected min_parameter getmin(int []com)

This function is a convenient comparator to get (return) the minimum value and the index of minimum of a int array" com []". This useful function is called in "Updateminsoct()".

protected boolean changed()

This Boolean function is used to determine whether the cost (link) is changed. And it's also called in "Updateminsoct()".

protected boolean Updateminsoct(int []temp,int id)

This function's name "Updateminsoct" is a mistake during our coding process, indeed, its name originally should be "Updatemincost", whatever, the name doesn't really matter. This Boolean function is also a judge function, using changed () and getmin() to determine whether to update the cost of node in "rtupdate()".

void linkhandler(int linkid, int newcost)

This function is called when cost from the node to link id changes from current value to new cost. This function calls rtupdate() to update the costs.

- Overall idea:

The main idea of this distributed asynchronous basic routing is that from time

to time, each node sends a copy of its distance vector to each of its neighbors. When a node x receives a new distance vector from any of its neighbors v, it saves v's distance vector, and uses Bellman-Ford to update its own distance vector:

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \text{ in } N$$

In our routines, rtupdate() uses values included in received routing packet to update nodes' own distance tables. According from DV algorithm, only directly connected nodes will exchange routing packets, which means nodes 1 and 2 will communicate with each other while 1 and 3 won't. During the whole process, the data structure, the 4-by-4 array of int [4][4] "costs", which called the distance table inside each node is our principal structure. And the routing table at each node thus can be directly derived from the distance table of this node.

Sample Outputs:

Using trace value of 2 and pseudorandom seed of 3322, and we get our sample outputs as follows:

```
Network Simulator v1.0
Enter trace level (>= 0): [0] 2
Enter seed (>= 0): [3322] 3322
                via
   D0 |    1     2     3
```

```
  ----|----------------
      1|     1    9999    9999
dest 2|  9999     10    9999
      3|  9999    9999      7
n0initialization complete!
                  via
   D1 |     0       2
  ----|----------------
      0|     1    9999
dest 2|  9999       1
      3|  9999    9999
n1initialization complete!
                  via
   D2 |     0       1     3
  ----|----------------
      0|    10    9999    9999
dest 1|  9999       1    9999
      3|  9999    9999       2
n2initialization complete!
                  via
   D3 |     0       2
  ----|----------------
```

```
      0|     7    9999
dest 1|   9999    9999
      2|   9999      2
n3initialization complete!
MAIN: rcv event, t=0.092, at 2 src: 0, dest: 2, contents:    0    1   10    7
********slience: n2
MAIN: rcv event, t=0.104, at 3 src: 0, dest: 3, contents:    0    1   10    7
MAIN: rcv event, t=0.115, at 1 src: 0, dest: 1, contents:    0    1   10    7
MAIN: rcv event, t=0.447, at 2 src: 1, dest: 2, contents:    1    0    1 9999
MAIN: rcv event, t=0.884, at 0 src: 1, dest: 0, contents:    1    0    1 9999
MAIN: rcv event, t=0.900, at 1 src: 2, dest: 1, contents:   10    1    0    2
MAIN: rcv event, t=1.048, at 1 src: 2, dest: 1, contents: 9999 9999    0    2
********slience: n1
MAIN: rcv event, t=2.032, at 3 src: 2, dest: 3, contents:   10    1    0    2
MAIN: rcv event, t=2.144, at 0 src: 2, dest: 0, contents:   10    1    0    2
********slience: n0
MAIN: rcv event, t=2.179, at 0 src: 3, dest: 0, contents:    7 9999    2    0
********slience: n0
MAIN: rcv event, t=2.284, at 0 src: 3, dest: 0, contents: 9999 9999    2    0
********slience: n0
MAIN: rcv event, t=2.306, at 2 src: 3, dest: 2, contents:    7 9999    2    0
********slience: n2
```

```
MAIN: rcv event, t=2.824, at 1 src: 0, dest: 1, contents:    0 9999 9999    7

********slience: n1

MAIN: rcv event, t=3.586, at 2 src: 3, dest: 2, contents:    7    8 9999    0

********slience: n2

MAIN: rcv event, t=3.658, at 3 src: 2, dest: 3, contents:    2    1    0 9999

MAIN: rcv event, t=3.680, at 0 src: 1, dest: 0, contents: 9999    0    1 9999

********slience: n0

MAIN: rcv event, t=3.976, at 0 src: 2, dest: 0, contents:    2    1    0    2

********slience: n0

MAIN: rcv event, t=4.268, at 2 src: 1, dest: 2, contents:    1    0 9999    8

********slience: n2

MAIN: rcv event, t=4.968, at 0 src: 1, dest: 0, contents: 9999    0    1    3

MAIN: rcv event, t=5.411, at 0 src: 3, dest: 0, contents: 9999    3    2    0

********slience: n0

MAIN: rcv event, t=5.524, at 3 src: 0, dest: 3, contents:    0    1    2 9999

********slience: n3

MAIN: rcv event, t=5.524, at 1 src: 0, dest: 1, contents:    0 9999 9999 9999

********slience: n1

MAIN: rcv event, t=5.543, at 2 src: 0, dest: 2, contents:    0    1    2    7

********slience: n2

MAIN: rcv event, t=6.266, at 0 src: 3, dest: 0, contents:    4    3    2    0

********slience: n0
```

```
MAIN: rcv event, t=6.446, at 3 src: 0, dest: 3, contents:    0    1    2    4

********slience: n3

MAIN: rcv event, t=6.708, at 2 src: 1, dest: 2, contents:    1    0 9999 9999

********slience: n2

MAIN: rcv event, t=7.918, at 2 src: 3, dest: 2, contents:    7 9999 9999    0

********slience: n2

MAIN: rcv event, t=8.463, at 2 src: 3, dest: 2, contents: 9999 9999 9999    0

********slience: n2

MAIN: rcv event, t=8.528, at 2 src: 0, dest: 2, contents:    0    1    2    4

********slience: n2

MAIN: rcv event, t=10000.000, at -1MAIN: rcv event, t=10000.619, at 2 src: 1, dest:
2, contents:   20    0 9999 9999

MAIN: rcv event, t=10000.723, at 0 src: 1, dest: 0, contents: 9999    0    1    3

********slience: n0

MAIN: rcv event, t=10000.806, at 0 src: 2, dest: 0, contents: 9999    1    0    2

********slience: n0

MAIN: rcv event, t=10001.366, at 3 src: 0, dest: 3, contents:    0 9999 9999 9999

********slience: n3

MAIN: rcv event, t=10001.877, at 1 src: 0, dest: 1, contents:    0   10    9    7

********slience: n1

MAIN: rcv event, t=10001.953, at 1 src: 2, dest: 1, contents:   10 9999    0    2

MAIN: rcv event, t=10002.117, at 2 src: 0, dest: 2, contents:    0   10    9    7
```

```
********slience: n2
MAIN: rcv event, t=10002.252, at 3 src: 2, dest: 3, contents:   10    1    0 9999
MAIN: rcv event, t=10003.089, at 2 src: 1, dest: 2, contents: 9999    0 9999 9999
********slience: n2
MAIN: rcv event, t=10003.356, at 0 src: 1, dest: 0, contents:   11    0    1    3
********slience: n0
MAIN: rcv event, t=10005.010, at 0 src: 3, dest: 0, contents: 9999    3    2    0
********slience: n0
MAIN: rcv event, t=10005.064, at 2 src: 3, dest: 2, contents:    7 9999 9999    0
MAIN: rcv event, t=10005.277, at 3 src: 2, dest: 3, contents: 9999    1    0 9999
********slience: n3
MAIN: rcv event, t=10006.062, at 0 src: 2, dest: 0, contents:    9    1    0    2
********slience: n0
MAIN: rcv event, t=10006.144, at 1 src: 2, dest: 1, contents:    9 9999    0    2
MAIN: rcv event, t=10006.326, at 0 src: 1, dest: 0, contents:   10    0    1    3
********slience: n0
MAIN: rcv event, t=10007.911, at 2 src: 1, dest: 2, contents: 9999    0 9999 9999
********slience: n2
MAIN: rcv event, t=20000.000, at -1**********************************
**********************************
**********************************
**********************************
```

```
**********************************

n0:0  n0:10  n0:9  n0:7

n1:10  n1:0  n1:1  n1:3

n2:9  n2:1  n2:0  n2:2

n3:7  n3:3  n3:2  n3:0

MAIN: rcv event, t=20000.025, at 1 src: 0, dest: 1, contents:    0 9999 9999 9999

********slience: n1

MAIN: rcv event, t=20000.107, at 3 src: 0, dest: 3, contents:    0    1    2    4

********slience: n3

MAIN: rcv event, t=20000.975, at 0 src: 1, dest: 0, contents: 9999    0    1    3

********slience: n0

MAIN: rcv event, t=20000.995, at 2 src: 0, dest: 2, contents:    0    1    2    4

********slience: n2

MAIN: rcv event, t=20001.851, at 2 src: 1, dest: 2, contents:    1    0 9999 9999

MAIN: rcv event, t=20002.484, at 3 src: 2, dest: 3, contents:    2    1    0 9999

MAIN: rcv event, t=20002.766, at 2 src: 3, dest: 2, contents: 9999 9999 9999    0

********slience: n2

MAIN: rcv event, t=20003.075, at 1 src: 2, dest: 1, contents: 9999 9999    0    2

********slience: n1

MAIN: rcv event, t=20003.253, at 0 src: 2, dest: 0, contents:    2    1    0    2

********slience: n0

MAIN: rcv event, t=20004.421, at 0 src: 3, dest: 0, contents:    4    3    2    0
```

********slience: n0

Simulator terminated at t=20004.420911, no packets in medium

Simulator terminated at time 20004.420911282694

****

The following matrix is the adjacency matrix of the topology(Not necessarily required!)

n0:0 n0:1 n0:2 n0:4

n1:1 n1:0 n1:1 n1:3

n2:2 n2:1 n2:0 n2:2

n3:4 n3:3 n3:2 n3:0

****

The following is the final distance table produced in each node:

via

D0	1	2	3
dest 1	1	11	10
dest 2	2	10	9
dest 3	4	12	7

via

D1	0	2

```
      0|      1     10000
dest 2|   10000         1
      3|   10000         3

                via

    D2 |      0       1      3
  ----|-----------------
      0|     10       2     10001
dest 1|     11       1     10001
      3|     14     10000       2

                via

    D3 |      0       2
  ----|-----------------
      0|      7       4
dest 1|      8       3
      2|      9       2
```

Part 2. With Poison Reverse (Prefinal Node)

Working correctness:

Each node initials correctly, and from node 0 to node 3, they all converge correctly with link changes and without link changes. And path vector works correctly.

Overall Designation:

- The extra functions different from basic routing:

void rtinit(int nodename, int[] initial_lkcost)

Initialize the data structure necessary for the routing algorithm, like routing table, two dimensional distance array. It also sends out messages to its direct neighbor to notify them its distance vector.

protected void rt_update()

Check whether the path is determined, if not, do not update the routing table.

protected boolean Need_change(int i)

Determine whether the new values we get are the same as the old one. If not, we have to update the routing table.

protected boolean Updateminsoct()

Update the two dimensional distance array and choose the minimum distance as a candidate.

protected boolean in_path(int nei,int des)

When sending distance vector to its neighbors, it must check whether the neighbor is within the path from it to a specific destination.

void linkhandler(int linkid, int newcost)

When the link weight changed, the node detects the change and recomputed the distance vector. If the routing table changed, it notifies its neighbors.

- The extra class:

public class table_ds

The data structure of routing table. It is a quadruple, it contains the destination, next hop, header and distance.

public class node_com implements Comparable<node_com>

This is a data structure in two dimensional distance table. For example, Dij stores the cost to a i via j, the header for this path and its index j.

- Overall idea:

The overall design is very similar to the poison reverse of the first part. But there are several different places. As to the data structure, the routing table and distance table must store the header value.

In the algorithm, when we update the distance table, we cannot simply choose the node with minimum value. Instead, we have to check whether all the nodes in its path go through this neighbor.

When sending distance vector to its neighbors, it must determine whether the neighbor is within its path to a specific destination instead of simply check whether the next hop is the neighbor.

The general procedure is:

(1) Initialize the node, send out messages to its neighbor.

(2) Every node keeps updating its routing table, the procedure is described above. Finally, every node converges to a specific value.

(3) Link cost changed. The nodes at two ends detect the change and update

its routing table. If table changes, send out its distance vector to neighbors. The procedure is the same as (2).

Sample Outputs:

Using trace value of 2 and pseudorandom seed of 3322, and we get our sample outputs as follows:

```
Network Simulator v1.0
Enter trace level (>= 0): [0] 2
Enter seed (>= 0): [3322] 3322
                via
   D0 |    1     2     3
  ----|-----------------
     1|    1  9999  9999
dest 2|  9999    10  9999
     3|  9999  9999     7
n0initialization complete!
                via
   D1 |    0     2
  ----|-----------------
     0|    1  9999
dest 2|  9999     1
     3|  9999  9999
```

```
n1initialization complete!
                    via
   D2 |     0       1      3
  ----|-----------------
      0|    10    9999    9999
dest 1|  9999       1    9999
      3|  9999    9999       2
n2initialization complete!
                    via
   D3 |     0       2
  ----|-----------------
      0|     7    9999
dest 1|  9999    9999
      2|  9999       2
n3initialization complete!
MAIN: rcv event, t=0.092, at 2 src: 0, dest: 2, contents:    0    1   10    7, headers:   -
1    0    0    0
********slience: n2
From 2 to 0: 2-〉0
From 2 to 1: 2-〉1
From 2 to 2:2
From 2 to 3: 2-〉3
```

```
MAIN: rcv event, t=0.104, at 3 src: 0, dest: 3, contents:   0   1  10   7, headers:  -
1   0   0   0
in_rt_update!
From 3 to 0: 3->0
From 3 to 1: 3->0->1
From 3 to 2: 3->2
From 3 to 3:3
MAIN: rcv event, t=0.115, at 1 src: 0, dest: 1, contents:   0   1  10   7, headers:  -
1   0   0   0
in_rt_update!
From 1 to 0: 1->0
From 1 to 1:1
From 1 to 2: 1->2
From 1 to 3: 1->0->3
MAIN: rcv event, t=0.447, at 2 src: 1, dest: 2, contents:   1   0   1 9999, headers:
1  -1   1  -1
in_rt_update!
From 2 to 0: 2->1->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
MAIN: rcv event, t=0.884, at 0 src: 1, dest: 0, contents:   1   0   1 9999, headers:
```

```
1  -1   1  -1

in_rt_update!

From 0 to 0:0

From 0 to 1: 0-〉1

From 0 to 2: 0-〉1-〉2

From 0 to 3: 0-〉3

MAIN: rcv event, t=0.900, at 1 src: 2, dest: 1, contents:  10   1   0   2, headers:

2   2  -1   2

in_rt_update!

From 1 to 0: 1-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=1.048, at 1 src: 2, dest: 1, contents: 9999 9999   0   2, headers:

-1  -1  -1   2

********slience: n1

From 1 to 0: 1-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=2.032, at 3 src: 2, dest: 3, contents:  10   1   0   2, headers:

2   2  -1   2
```

```
in_rt_update!
From 3 to 0: 3->0
From 3 to 1: 3->2->1
From 3 to 2: 3->2
From 3 to 3:3
MAIN: rcv event, t=2.144, at 0 src: 2, dest: 0, contents:   10    1    0    2, headers:
2    2   -1    2
********slience: n0
From 0 to 0:0
From 0 to 1: 0->1
From 0 to 2: 0->1->2
From 0 to 3: 0->3
MAIN: rcv event, t=2.179, at 0 src: 3, dest: 0, contents:    7 9999    2    0, headers:
3   -1    3   -1
********slience: n0
From 0 to 0:0
From 0 to 1: 0->1
From 0 to 2: 0->1->2
From 0 to 3: 0->3
MAIN: rcv event, t=2.284, at 0 src: 3, dest: 0, contents: 9999 9999    2    0, headers:
-1   -1    3   -1
********slience: n0
```

```
From 0 to 0:0

From 0 to 1: 0-〉1

From 0 to 2: 0-〉1-〉2

From 0 to 3: 0-〉3

MAIN: rcv event, t=2.306, at 2 src: 3, dest: 2, contents:    7 9999    2    0, headers:

3   -1    3   -1

********slience: n2

From 2 to 0: 2-〉1-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3

MAIN: rcv event, t=2.824, at 1 src: 0, dest: 1, contents:    0 9999 9999    7, headers:

-1   -1   -1    0

********slience: n1

From 1 to 0: 1-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=3.586, at 2 src: 3, dest: 2, contents:    7    8 9999    0, headers:

3    0   -1   -1

********slience: n2

From 2 to 0: 2-〉1-〉0
```

```
From 2 to 1: 2->1

From 2 to 2:2

From 2 to 3: 2->3

MAIN: rcv event, t=3.658, at 3 src: 2, dest: 3, contents:    2    1    0 9999, headers:

1    2   -1   -1

in_rt_update!

From 3 to 0: 3->2->1->0

From 3 to 1: 3->2->1

From 3 to 2: 3->2

From 3 to 3:3

MAIN: rcv event, t=3.680, at 0 src: 1, dest: 0, contents: 9999    0    1 9999, headers:

-1   -1    1   -1

********slience: n0

From 0 to 0:0

From 0 to 1: 0->1

From 0 to 2: 0->1->2

From 0 to 3: 0->3

MAIN: rcv event, t=3.976, at 0 src: 2, dest: 0, contents: 9999    1    0    2, headers:

-1    2   -1    2

********slience: n0

From 0 to 0:0

From 0 to 1: 0->1
```

```
From 0 to 2: 0->1->2

From 0 to 3: 0->3

MAIN: rcv event, t=4.268, at 2 src: 1, dest: 2, contents:    1    0 9999    8, headers:

1   -1   -1    0

********slience: n2

From 2 to 0: 2->1->0

From 2 to 1: 2->1

From 2 to 2:2

From 2 to 3: 2->3

MAIN: rcv event, t=4.968, at 0 src: 1, dest: 0, contents: 9999    0    1    3, headers:

-1   -1    1    2

in_rt_update!

From 0 to 0:0

From 0 to 1: 0->1

From 0 to 2: 0->1->2

From 0 to 3: 0->1->2->3

MAIN: rcv event, t=5.411, at 0 src: 3, dest: 0, contents: 9999    3    2    0, headers:

-1    2    3   -1

********slience: n0

From 0 to 0:0

From 0 to 1: 0->1

From 0 to 2: 0->1->2
```

```
From 0 to 3: 0-〉1-〉2-〉3

MAIN: rcv event, t=5.524, at 3 src: 0, dest: 3, contents:   0   1   2 9999, headers:

-1   0   1  -1

********slience: n3

From 3 to 0: 3-〉2-〉1-〉0

From 3 to 1: 3-〉2-〉1

From 3 to 2: 3-〉2

From 3 to 3:3

MAIN: rcv event, t=5.524, at 1 src: 0, dest: 1, contents:   0 9999 9999 9999,

headers:  -1  -1  -1  -1

********slience: n1

From 1 to 0: 1-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=5.543, at 2 src: 0, dest: 2, contents:   0   1 9999   7, headers:

-1   0  -1   0

********slience: n2

From 2 to 0: 2-〉1-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3
```

```
MAIN: rcv event, t=6.266, at 0 src: 3, dest: 0, contents: 9999    3    2    0, headers:
-1    2    3   -1
********slience: n0
From 0 to 0:0
From 0 to 1: 0-〉1
From 0 to 2: 0-〉1-〉2
From 0 to 3: 0-〉1-〉2-〉3
MAIN: rcv event, t=6.446, at 3 src: 0, dest: 3, contents:    0    1    2 9999, headers:
-1    0    1   -1
********slience: n3
From 3 to 0: 3-〉2-〉1-〉0
From 3 to 1: 3-〉2-〉1
From 3 to 2: 3-〉2
From 3 to 3:3
MAIN: rcv event, t=6.708, at 2 src: 1, dest: 2, contents:    1    0 9999 9999, headers:
1   -1   -1   -1
********slience: n2
From 2 to 0: 2-〉1-〉0
From 2 to 1: 2-〉1
From 2 to 2:2
From 2 to 3: 2-〉3
MAIN: rcv event, t=7.918, at 2 src: 3, dest: 2, contents:    7 9999 9999    0, headers:
```

```
3  -1  -1  -1

********slience: n2

From 2 to 0: 2-〉1-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3

MAIN: rcv event, t=8.463, at 2 src: 3, dest: 2, contents: 9999 9999 9999    0,

headers:  -1  -1  -1  -1

********slience: n2

From 2 to 0: 2-〉1-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3

MAIN: rcv event, t=8.528, at 2 src: 0, dest: 2, contents:    0    1 9999 9999, headers:

-1    0  -1  -1

********slience: n2

From 2 to 0: 2-〉1-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3

MAIN: rcv event, t=10000.000, at -1in_rt_update!

From 1 to 0: 1-〉0
```

```
From 1 to 1:1
From 1 to 2: 1->2
From 1 to 3: 1->2->3
in_rt_update!
From 0 to 0:0
From 0 to 1: 0->3->2->1
From 0 to 2: 0->3->2
From 0 to 3: 0->3
MAIN: rcv event, t=10000.619, at 2 src: 1, dest: 2, contents:   20    0 9999 9999,
headers:    1   -1   -1   -1
in_rt_update!
From 2 to 0: 2->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
MAIN: rcv event, t=10000.723, at 0 src: 1, dest: 0, contents: 9999    0    1    3,
headers:   -1   -1    1    2
********slience: n0
From 0 to 0:0
From 0 to 1: 0->3->2->1
From 0 to 2: 0->3->2
From 0 to 3: 0->3
```

```
MAIN: rcv event, t=10000.806, at 0 src: 2, dest: 0, contents: 9999  1  0  2,
headers:  -1  2  -1  2
********slience: n0
From 0 to 0:0
From 0 to 1: 0->3->2->1
From 0 to 2: 0->3->2
From 0 to 3: 0->3
MAIN: rcv event, t=10001.366, at 3 src: 0, dest: 3, contents:  0 9999 9999 9999,
headers:  -1  -1  -1  -1
********slience: n3
From 3 to 0: 3->2->1->0
From 3 to 1: 3->2->1
From 3 to 2: 3->2
From 3 to 3:3
MAIN: rcv event, t=10001.877, at 1 src: 0, dest: 1, contents:  0 9999  9  7,
headers:  -1  -1  3  0
********slience: n1
From 1 to 0: 1->0
From 1 to 1:1
From 1 to 2: 1->2
From 1 to 3: 1->2->3
MAIN: rcv event, t=10001.953, at 1 src: 2, dest: 1, contents:  10 9999  0  2,
```

```
headers:    2   -1   -1    2

in_rt_update!

From 1 to 0: 1-〉2-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=10002.117, at 2 src: 0, dest: 2, contents:    0 9999 9999    7,

headers:   -1   -1   -1    0

********slience: n2

From 2 to 0: 2-〉0

From 2 to 1: 2-〉1

From 2 to 2:2

From 2 to 3: 2-〉3

MAIN: rcv event, t=10002.252, at 3 src: 2, dest: 3, contents:   10    1    0 9999,

headers:    2    2   -1   -1

in_rt_update!

From 3 to 0: 3-〉0

From 3 to 1: 3-〉2-〉1

From 3 to 2: 3-〉2

From 3 to 3:3

MAIN: rcv event, t=10003.089, at 2 src: 1, dest: 2, contents: 9999    0 9999 9999,

headers:   -1   -1   -1   -1
```

```
********slience: n2
From 2 to 0: 2-〉0
From 2 to 1: 2-〉1
From 2 to 2:2
From 2 to 3: 2-〉3
MAIN: rcv event, t=10003.356, at 0 src: 1, dest: 0, contents: 9999    0    1    3,
headers:  -1  -1    1    2
********slience: n0
From 0 to 0:0
From 0 to 1: 0-〉3-〉2-〉1
From 0 to 2: 0-〉3-〉2
From 0 to 3: 0-〉3
MAIN: rcv event, t=10005.010, at 0 src: 3, dest: 0, contents: 9999    3    2    0,
headers:  -1    2    3  -1
********slience: n0
From 0 to 0:0
From 0 to 1: 0-〉3-〉2-〉1
From 0 to 2: 0-〉3-〉2
From 0 to 3: 0-〉3
MAIN: rcv event, t=10005.064, at 2 src: 3, dest: 2, contents:    7 9999 9999    0,
headers:    3  -1  -1  -1
in_rt_update!
```

```
From 2 to 0: 2->3->0

From 2 to 1: 2->1

From 2 to 2:2

From 2 to 3: 2->3

MAIN: rcv event, t=10005.277, at 3 src: 2, dest: 3, contents: 9999    1    0 9999,

headers:  -1    2  -1  -1

********slience: n3

From 3 to 0: 3->0

From 3 to 1: 3->2->1

From 3 to 2: 3->2

From 3 to 3:3

MAIN: rcv event, t=10006.062, at 0 src: 2, dest: 0, contents: 9999    1    0    2,

headers:  -1    2  -1    2

********slience: n0

From 0 to 0:0

From 0 to 1: 0->3->2->1

From 0 to 2: 0->3->2

From 0 to 3: 0->3

MAIN: rcv event, t=10006.144, at 1 src: 2, dest: 1, contents:    9 9999    0    2,

headers:    3  -1  -1    2

in_rt_update!

From 1 to 0: 1->2->3->0
```

```
From 1 to 1:1
From 1 to 2: 1->2
From 1 to 3: 1->2->3
MAIN: rcv event, t=10006.326, at 0 src: 1, dest: 0, contents: 9999  0  1  3,
headers:  -1  -1  1  2
********slience: n0
From 0 to 0:0
From 0 to 1: 0->3->2->1
From 0 to 2: 0->3->2
From 0 to 3: 0->3
MAIN: rcv event, t=10007.911, at 2 src: 1, dest: 2, contents: 9999  0 9999 9999,
headers:  -1  -1  -1  -1
********slience: n2
From 2 to 0: 2->3->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
MAIN: rcv event, t=20000.000, at -1
**********************************
**********************************
**********************************
**********************************
```

```
**********************************


n0:0  n0:10  n0:9  n0:7

n1:10  n1:0  n1:1  n1:3

n2:9  n2:1  n2:0  n2:2

n3:7  n3:3  n3:2  n3:0

in_rt_update!

From 0 to 0:0

From 0 to 1: 0->1

From 0 to 2: 0->1->2

From 0 to 3: 0->1->2->3

in_rt_update!

From 1 to 0: 1->0

From 1 to 1:1

From 1 to 2: 1->2

From 1 to 3: 1->2->3

MAIN: rcv event, t=20000.025, at 1 src: 0, dest: 1, contents:    0 9999 9999 9999,

headers:  -1  -1  -1  -1

********slience: n1

From 1 to 0: 1->0

From 1 to 1:1

From 1 to 2: 1->2
```

```
From 1 to 3: 1->2->3
MAIN: rcv event, t=20000.107, at 3 src: 0, dest: 3, contents:   0   1   2 9999,
headers:  -1   0   1  -1
********slience: n3
From 3 to 0: 3->0
From 3 to 1: 3->2->1
From 3 to 2: 3->2
From 3 to 3:3
MAIN: rcv event, t=20000.975, at 0 src: 1, dest: 0, contents: 9999   0   1   3,
headers:  -1  -1   1   2
********slience: n0
From 0 to 0:0
From 0 to 1: 0->1
From 0 to 2: 0->1->2
From 0 to 3: 0->1->2->3
MAIN: rcv event, t=20000.995, at 2 src: 0, dest: 2, contents:   0   1 9999 9999,
headers:  -1   0  -1  -1
********slience: n2
From 2 to 0: 2->3->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
```

```
MAIN: rcv event, t=20001.851, at 2 src: 1, dest: 2, contents:    1    0 9999 9999,
headers:    1   -1   -1   -1
in_rt_update!
From 2 to 0: 2->1->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
MAIN: rcv event, t=20002.484, at 3 src: 2, dest: 3, contents:    2    1    0 9999,
headers:    1    2   -1   -1
in_rt_update!
From 3 to 0: 3->2->1->0
From 3 to 1: 3->2->1
From 3 to 2: 3->2
From 3 to 3:3
MAIN: rcv event, t=20002.766, at 2 src: 3, dest: 2, contents: 9999 9999 9999    0,
headers:   -1   -1   -1   -1
********slience: n2
From 2 to 0: 2->1->0
From 2 to 1: 2->1
From 2 to 2:2
From 2 to 3: 2->3
MAIN: rcv event, t=20003.075, at 1 src: 2, dest: 1, contents: 9999 9999    0    2,
```

```
headers:  -1  -1  -1   2

********slience: n1

From 1 to 0: 1-〉0

From 1 to 1:1

From 1 to 2: 1-〉2

From 1 to 3: 1-〉2-〉3

MAIN: rcv event, t=20003.253, at 0 src: 2, dest: 0, contents: 9999   1   0   2,

headers:  -1   2  -1   2

********slience: n0

From 0 to 0:0

From 0 to 1: 0-〉1

From 0 to 2: 0-〉1-〉2

From 0 to 3: 0-〉1-〉2-〉3

MAIN: rcv event, t=20004.421, at 0 src: 3, dest: 0, contents: 9999   3   2   0,

headers:  -1   2   3  -1

********slience: n0

From 0 to 0:0

From 0 to 1: 0-〉1

From 0 to 2: 0-〉1-〉2

From 0 to 3: 0-〉1-〉2-〉3


Simulator terminated at t=20004.420911, no packets in medium
```

Simulator terminated at time 20004.420911282694

****

The following matrix is the adjacency matrix of the topology(Not necessarily required!)

n0:0 n0:1 n0:2 n0:4

n1:1 n1:0 n1:1 n1:3

n2:2 n2:1 n2:0 n2:2

n3:4 n3:3 n3:2 n3:0

****

The following is the final distance table produced in each node:

via

D0	1	2	3
1	1	11	10
dest 2	2	10	9
3	4	12	7

via

D1	0	2
0	1	9999
dest 2	9999	1
3	9999	3

via

D2	0	1	3
0	10	2	9999
dest 1	11	1	9999
3	9999	9999	2

via

D3	0	2
0	7	4
dest 1	8	3
2	9	2