Programming Assignment 2

Amanda Cadwell-Frost

21 November, 2011

Design

"How It Works"

It should be noted that I got confused as I wrote this assignment, and reversed the i and the j in distanceTable[i][j]. However, since this error is consistent throughout the program, it still runs smoothly.

The driving force behind this routing simulation is the distance-vector algorithm. This is a distributed routing algorithm, where each node calculates the distance from itself to its 1-hop neighbors, and then broadcasts that information. Eventually, every router in a network has calculated the shortest path from itself to every other router.

In the initialization phase, I hardcoded the values given by the distance diagram into the distance table. This is a slight design flaw, I could have used the public cost array. However, the assignment does not have to be agile in this way. I also set all of the unknown costs to 999, to signify that they were unknown. Next, I used the new Distance Table to create an array, minDistance, which held the minimum costs to each neighbor. Finally, after each Entity is initialized, it broadcasts this information to all of its neighbors.

Upon receiving a packet holding a minimum cost array, each entity updates its distance table, so that each value that depended on reaching a node, n, via the source of the packet was equal to the new value of reaching n plus the minimum cost of moving from the receiving entity to the source of the packet. Next, it recalculated the minimums, and, if any of them changed, the Entity again broadcast an update to each of its neighbors. If the minimums are unchanged, the Entity does not rebroadcast, which ensures that the program will eventually stop.

I would really have liked to have made my program agile, so that it could update itself if the cost of any of the links changed. In order to do this, when a change appeared, each affected Entity would reset all of its costs to 999, except the ones to its direct neighbors, which would be as original. Then, I would change the value of the changed link to newCost, and send out an update packet.

Ideally, the other neighbors, upon receiving this packet, would change the value of their distance tables to reflect the new costs. However, my method for finding the minimum values meant that the higher costs were never reflected.

As I wrote this, I suddenly had an epiphany, and changed my code to reflect it. So the extra credit is now implemented.

I would still like to change my code to make it so that fewer numbers are hard coded, and so that I could just directly copy and past the code from one to another without having to change more than one line.

A way in which my code does not work is that, as I run it, the link does not change back to 1 at time 20000. In fact, the program kicks out and ends at about time 10014. I don't think that this is an error in my code, however, so I am unconcerned.

Testing

I will be the first to admit that this code is not exactly well-tested. I did not attempt to run it with different data than that which is shown in the diagram. I did go through the diagram and check that each distance output given was correct, so I know that for the given case the code is complete.

I also tested it with the extra credit data, and verified that each resulting number was correct.

Therefore, I am confident that my code completely fulfills the requirements of this assignment, and therefore I would be willing to say that it should work on any given set of data, but I admit that I cannot be 100 percent certain that it will run properly in all cases.

The fact that when I run the code, the event of the cost from 0 to 1 returning to 1 does not occur, is worrying. I am not sure why this is happening, and I do not believe that it is an error on my part.

Code

As I completed this project, I changed the constructor and the update class in each of Entity0.java, Entity1.java, Entity2.java, and Entity3.java. I commented Entity0.java very thoroughly, and then only commented those places in the other three classes where they differed from Entity0.java.

Following is the code for Entity0.java.

Constructor Class

```
 public Entity0()
   {
     //This for loop initializes all of the costs to infinity, because no values are known.
     for(int f = 0; f < 4; f++){
       for(int g = 0; g < 4; g++){
         distanceTable[f][g] = 999;
       }
     }
     //we can now plug in the costs that are given in the diagram
     distanceTable[0][0] = 0;
     distanceTable[1][1] = 1;
     distanceTable[2][2] = 2;
     distanceTable[3][3] = 7;
     //This creates a new array which we will use to hold Entity0's min costs to each other node,
     //and eventually send that data using toLayer2().
     int[] minDistance = new int[4];
     //This for loop actually calculates the minimum cost to each node.
     for(int h = 0; h < 4; h++){
       //int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
       int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
       minDistance[h] = Math.min(distanceTable[h][1], b);
     }
     //This for loop sends the packet containing cost data to each other node.  It starts at 1 to avoid
```

```
  //sending the packet to Entity0.
  for(int i = 1; i < 4; i++){
    //this line builds the packet with destination 1-3
    Packet dtPacket = new Packet(0, i, minDistance);
    //this line sends it.
    NetworkSimulator.toLayer2(dtPacket);
  }
  // Prints the time, the fact that the initialization has completed, and the current Distance Table.
  System.out.println("Entity0 Initializion Complete. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}

public void update(Packet p)
{
  //first, we define a boolean to mark whether any minDistance has changed. If none have changed,
  //we won't send out a packet.  It's initialized to false, because nothing has changed yet.
  boolean send = false;
  //Again, an array to hold and eventually send minimum distance values
  int[] minDistance = new int[4];
  //This for loop recalculates the min costs to each node and again store them in minDistance.
  for(int h = 0; h < 4; h++){
    //int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
    int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(distanceTable[h][1], b);
  }
  //This for loop checks to see if the cost in the distance table of getting to each node via p's source,
  // is less than the cost found by calculating the Mincost given by p plus the cost in the distance table
  //of getting to each node.
  for(int k = 0; k<4; k++){
    if(p.getMincost(k)+minDistance[p.getSource()] < distanceTable[k][p.getSource()]){
      //if so, it changes the value in the distance table to reflect the lesser value
      distanceTable[k][p.getSource()] = p.getMincost(k)+minDistance[p.getSource()];
      //it also checks to see if this is the new minimum distance for reaching this node.  If this
      //minimum value has changed, it changes send to true.
      if(distanceTable[k][p.getSource()]<minDistance[k]){
        minDistance[k] = distanceTable[k][p.getSource()];
        send = true;
      }
    }
  }
  //if the minimum value has changed, it sends update packets to each other node.
  if(send){
    for(int i = 1; i < 4; i++){
      Packet dtPacket = new Packet(0, i, minDistance);
      NetworkSimulator.toLayer2(dtPacket);
    }
  }
  //Prints the time, the fac that the update method just completed, and the current Distance Table.
  System.out.println("Entity0 Update Complete. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}
```

```
public void linkCostChangeHandler(int whichLink, int newCost)
{
  //we start out by deleting all previous data, becuase it could be wrong.
  for(int f = 0; f < 4; f++){
    for(int g = 0; g < 4; g++){
      distanceTable[f][g] = 999;
    }
  }
  //we can now plug in the costs that are given in the diagram
  distanceTable[0][0] = 0;
  distanceTable[1][1] = 1;
  distanceTable[2][2] = 2;
  distanceTable[3][3] = 7;
  //next, we replace the value of getting to the node with the changed link with the new cost
  distanceTable[whichLink][whichLink] = newCost;
  //Once again, we initialize and calculate the minimum distance array.
  int[] minDistance = new int[4];
  for(int h = 0; h < 4; h++){
    int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(distanceTable[h][1], b);
  }
  //finally, we send packets out to each node with the new costs.
  for(int i = 1; i < 4; i++){
    Packet dtPacket = new Packet(0, i, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
  }
  System.out.println("Entity0 linkCostChange Handled. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}
```

Entity1.java

```
public Entity1()
{
  for(int f = 0; f < 4; f++){
    for(int g = 0; g < 4; g++){
      distanceTable[f][g] = 999;
    }
  }
  //This section has the same function as in Entity0, but the values are Entity1's.  As you can see,
  //Entity1 does not connected to Entity3, so the cost between them is infinite.
  distanceTable[0][0] = 1;
  distanceTable[1][1] = 0;
  distanceTable[2][2] = 1;
  distanceTable[3][3] = 999;
  int[] minDistance = new int[4];
  for(int h = 0; h < 4; h++){
    //int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
    int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(distanceTable[h][0], b);
  }
```

```
    //This send protocol differs from Entity0's, because Entity1 only sends 2 packets, and I decided that it
would
    //be more efficient if I just made each packet than if I were to run a for loop.
    Packet dtPacket = new Packet(1, 0, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
    dtPacket = new Packet(1, 2, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
    System.out.println("Entity1 Initializion Complete. Distance Table is:");
    System.out.println("Time now is " + NetworkSimulator.time + ".");
    printDT();
  }

  public void update(Packet p)
  {
    boolean send = false;
    int[] minDistance = new int[4];
    for(int h = 0; h < 4; h++){
      //int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
      int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
      minDistance[h] = Math.min(distanceTable[h][0], b);
    }
    for(int k = 0; k<4; k++){
      if(p.getMincost(k)+minDistance[p.getSource()] < distanceTable[k][p.getSource()]){
        distanceTable[k][p.getSource()] = p.getMincost(k)+minDistance[p.getSource()];
        if(distanceTable[k][p.getSource()]<minDistance[k]){
          minDistance[k] = distanceTable[k][p.getSource()];
          send = true;
        }
      }
    }
    //this send protocol is the same as the one in the construstor.  It once again differs from Entity0's.
    if(send){
      Packet dtPacket = new Packet(1, 0, minDistance);
      NetworkSimulator.toLayer2(dtPacket);
      dtPacket = new Packet(1, 2, minDistance);
      NetworkSimulator.toLayer2(dtPacket);
    }
    System.out.println("Entity1 Update Complete. Distance Table is:");
    System.out.println("Time now is " + NetworkSimulator.time + ".");
    printDT();
  }

  public void linkCostChangeHandler(int whichLink, int newCost)
  {
    //we start out by deleting all previous data, becuase it could be wrong.
    for(int f = 0; f < 4; f++){
      for(int g = 0; g < 4; g++){
        distanceTable[f][g] = 999;
      }
    }
    //we can now plug in the costs that are given in the diagram
    distanceTable[0][0] = 1;
    distanceTable[1][1] = 0;
    distanceTable[2][2] = 1;
    distanceTable[3][3] = 999;
    //next, we replace the value of getting to the node with the changed link with the new cost
    distanceTable[whichLink][whichLink] = newCost;
    //Once again, we initialize and calculate the minimum distance array.
    int[] minDistance = new int[4];
    for(int h = 0; h < 4; h++){
      int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
      minDistance[h] = Math.min(distanceTable[h][0], b);
    }
    //finally, we send packets out to each node with the new costs.
    Packet dtPacket = new Packet(1, 0, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
    dtPacket = new Packet(1, 2, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
    System.out.println("Entity1 linkCostChange Handled. Distance Table is:");
    System.out.println("Time now is " + NetworkSimulator.time + ".");
    printDT();
  }
```

Entity2.java

```
    public Entity2()
  {
    for(int f = 0; f < 4; f++){
      for(int g = 0; g < 4; g++){
        distanceTable[f][g] = 999;
      }
    }
    //this is one of thep places where Entity1 differs.  The distance table is being initialized with values
    //pertaining to Entity2, not Entity0.
    distanceTable[0][0] = 3;
    distanceTable[1][1] = 1;
    distanceTable[2][2] = 0;
    distanceTable[3][3] = 2;

    int[] minDistance = new int[4];
    for(int h = 0; h < 4; h++){
      int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
      minDistance[h] = Math.min(a, distanceTable[h][3]);
    }
    //This is another place where Entity2 differs.  As you can see, my for loop loops through all four
entities,
    //but checks to make sure that it doesn't actually send a packet to itself using an if.
    //Also, the from field when making the packet is 2.
    for(int i = 0; i < 4; i++){
      if(i != 2){
        Packet dtPacket = new Packet(2, i, minDistance);
        NetworkSimulator.toLayer2(dtPacket);
      }
    }
    System.out.println("Entity2 Initializion Complete. Distance Table is:");
    System.out.println("Time now is " + NetworkSimulator.time + ".");
    printDT();
```

```
}

public void update(Packet p)
{
  boolean send = false;
  int[] minDistance = new int[4];
  for(int h = 0; h < 4; h++){
    int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
    int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(a, b);
  }
  for(int k = 0; k<4; k++){
    if(p.getMincost(k)+minDistance[p.getSource()] < distanceTable[k][p.getSource()]){
      distanceTable[k][p.getSource()] = p.getMincost(k)+minDistance[p.getSource()];
      if(distanceTable[k][p.getSource()]<minDistance[k]){
        minDistance[k] = distanceTable[k][p.getSource()];
        send = true;
      }
    }
  }
  //again, the sending protocol is different.
  if(send){
    for(int i = 0; i < 4; i++){
      if(i != 2){
        Packet dtPacket = new Packet(2, i, minDistance);
        NetworkSimulator.toLayer2(dtPacket);
      }
    }
  }
  System.out.println("Entity2 Update Complete. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}

public void linkCostChangeHandler(int whichLink, int newCost)
{
  //we start out by deleting all previous data, becuase it could be wrong.
  for(int f = 0; f < 4; f++){
    for(int g = 0; g < 4; g++){
      distanceTable[f][g] = 999;
    }
  }
  //we can now plug in the costs that are given in the diagram
  distanceTable[0][0] = 3;
  distanceTable[1][1] = 1;
  distanceTable[2][2] = 0;
  distanceTable[3][3] = 2;
  //next, we replace the value of getting to the node with the changed link with the new cost
  distanceTable[whichLink][whichLink] = newCost;
  //Once again, we initialize and calculate the minimum distance array.
  int[] minDistance = new int[4];
  for(int h = 0; h < 4; h++){
    int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
    //int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
```

```
        minDistance[h] = Math.min(a, distanceTable[h][3]);
      }
      //finally, we send packets out to each node with the new costs.
      for(int i = 0; i < 4; i++){
        if(i != 2){
          Packet dtPacket = new Packet(2, i, minDistance);
          NetworkSimulator.toLayer2(dtPacket);
        }
      }
      System.out.println("Entity2 linkCostChange Handled. Distance Table is:");
      System.out.println("Time now is " + NetworkSimulator.time + ".");
      printDT();
    }
```

Entity3.java

```
    public Entity3()
    {
      for(int f = 0; f < 4; f++){
        for(int g = 0; g < 4; g++){
          distanceTable[f][g] = 999;
        }
      }
      //as with Entities 1 and 2, the distance table is being filled with Entity3's values.
      distanceTable[0][0] = 7;
      distanceTable[1][1] = 999;
      distanceTable[2][2] = 2;
      distanceTable[3][3] = 0;
      int[] minDistance = new int[4];
      for(int h = 0; h < 4; h++){
        int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
        int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
        minDistance[h] = Math.min(a, b);
      }
      //as with Entity1, Entity3 only sends 2 packets (because it does not link to Entity1), so i chose to make
them
      //individually instead of using a for loop.  Also, the from field is filled in with 3.
      Packet dtPacket = new Packet(3, 0, minDistance);
      NetworkSimulator.toLayer2(dtPacket);
      dtPacket = new Packet(3, 2, minDistance);
      NetworkSimulator.toLayer2(dtPacket);
      System.out.println("Entity3 Initializion Complete. Distance Table is:");
      System.out.println("Time now is " + NetworkSimulator.time + ".");
      printDT();
    }

    public void update(Packet p)
    {
      boolean send = false;
      int[] minDistance = new int[4];
      for(int h = 0; h < 4; h++){
        int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
        //int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(a, distanceTable[h][2]);
  }
  for(int k = 0; k<4; k++){
    if(p.getMincost(k)+minDistance[p.getSource()] < distanceTable[k][p.getSource()]){
      distanceTable[k][p.getSource()] = p.getMincost(k)+minDistance[p.getSource()];
      if(distanceTable[k][p.getSource()]<minDistance[k]){
        minDistance[k] = distanceTable[k][p.getSource()];
        send = true;
      }
    }
  }
  //Once again, the sending is different.
  if(send){
    Packet dtPacket = new Packet(3, 0, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
    dtPacket = new Packet(3, 2, minDistance);
    NetworkSimulator.toLayer2(dtPacket);
  }
  System.out.println("Entity3 Update Complete. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}

public void linkCostChangeHandler(int whichLink, int newCost)
{
  //we start out by deleting all previous data, becuase it could be wrong.
  for(int f = 0; f < 4; f++){
    for(int g = 0; g < 4; g++){
      distanceTable[f][g] = 999;
    }
  }
  //we can now plug in the costs that are given in the diagram
  distanceTable[0][0] = 7;
  distanceTable[1][1] = 999;
  distanceTable[2][2] = 2;
  distanceTable[3][3] = 0;
  //next, we replace the value of getting to the node with the changed link with the new cost
  distanceTable[whichLink][whichLink] = newCost;
  //Once again, we initialize and calculate the minimum distance array.
  int[] minDistance = new int[4];
  for(int h = 0; h < 4; h++){
    int a = Math.min(distanceTable[h][0], distanceTable[h][1]);
    //int b = Math.min(distanceTable[h][2], distanceTable[h][3]);
    minDistance[h] = Math.min(a, distanceTable[h][2]);
  }
  //finally, we send packets out to each node with the new costs.
  Packet dtPacket = new Packet(3, 0, minDistance);
  NetworkSimulator.toLayer2(dtPacket);
  dtPacket = new Packet(3, 2, minDistance);
  NetworkSimulator.toLayer2(dtPacket);
  System.out.println("Entity3 linkCostChange Handled. Distance Table is:");
  System.out.println("Time now is " + NetworkSimulator.time + ".");
  printDT();
}
```

Sample Output

```
> run Project
Network Simulator v1.0
Enter trace level (>= 0): [0]  0
Will the link change (1 = Yes, 0 = No): [0]  1
Enter random seed: [random]  4317298
Entity0 Initializion Complete. Distance Table is:
Time now is 0.0.

          via
 D0 l   1   2   3
----+------------
   1l   1 999 999
   2l 999   2 999
   3l 999 999   7
Entity1 Initializion Complete. Distance Table is:
Time now is 0.0.

        via
 D1 l   0   2
----+--------
   0l   1 999
   2l 999   1
   3l 999 999
Entity2 Initializion Complete. Distance Table is:
Time now is 0.0.

          via
 D2 l   0   1   3
----+------------
   0l   3 999 999
   1l 999   1 999
   3l 999 999   2
Entity3 Initializion Complete. Distance Table is:
Time now is 0.0.
        via
 D3 l   0   2
----+--------
   0l   7 999
   1l 999 999
   2l 999   2
Entity3 Update Complete. Distance Table is:
Time now is 1.002179309514022.
        via
 D3 l   0   2
----+--------
   0l   7 999
   1l   8 999
   2l   9   2
Entity2 Update Complete. Distance Table is:
```

```
Time now is 3.6711012808118957.

          via
 D2 l   0   1   3
----+------------
   0l   3 999 999
   1l   4   1 999
   3l  10 999   2
Entity3 Update Complete. Distance Table is:
Time now is 3.7129146385058185.
        via
 D3 l   0   2
----+--------
   0l   7   5
   1l   8   3
   2l   9   2
Entity0 Update Complete. Distance Table is:
Time now is 4.824110642938916.

          via
 D0 l   1   2   3
----+------------
   1l   1 999 999
   2l   2   2 999
   3l 999 999   7
Entity0 Update Complete. Distance Table is:
Time now is 5.518225331524891.

          via
 D0 l   1   2   3
----+------------
   1l   1   3 999
   2l   2   2 999
   3l 999   4   7
Entity2 Update Complete. Distance Table is:
Time now is 6.459331812124074.

          via
 D2 l   0   1   3
----+------------
   0l   3   2 999
   1l   4   1 999
   3l  10 999   2
Entity0 Update Complete. Distance Table is:
Time now is 7.2216148616622124.

          via
 D0 l   1   2   3
----+------------
   1l   1   3 999
   2l   2   2   6
   3l 999   4   4
Entity1 Update Complete. Distance Table is:
Time now is 7.364781344643764.
```

```
         via
 D1 l   0   2
----+--------
   0l   1   4
   2l 999   1
   3l 999   3
Entity1 Update Complete. Distance Table is:
Time now is 7.538346727075317.

         via
 D1 l   0   2
----+--------
   0l   1   4
   2l   3   1
   3l   8   3
Entity2 Update Complete. Distance Table is:
Time now is 8.93011233964956.

           via
 D2 l   0   1   3
----+------------
   0l   3   2 999
   1l   3   1 999
   3l   9 999   2
Entity0 Update Complete. Distance Table is:
Time now is 9.224186743116778.

           via
 D0 l   1   2   3
----+------------
   1l   1   3 999
   2l   2   2   6
   3l   4   4   4
Entity2 Update Complete. Distance Table is:
Time now is 9.378557269589258.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   9
   1l   3   1 999
   3l   9 999   2
Entity2 Update Complete. Distance Table is:
Time now is 10.328517344309438.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   9
   1l   3   1 999
   3l   6 999   2
Entity3 Update Complete. Distance Table is:
Time now is 10.329243899327148.
```

```
        via
 D3 l   0   2
----+--------
   0l   7   5
   1l   6   3
   2l   7   2
Entity2 Update Complete. Distance Table is:
Time now is 11.425553262826927.

          via
 D2 l   0   1   3
----+------------
   0l   3   2   9
   1l   3   1 999
   3l   6   4   2
Entity1 Update Complete. Distance Table is:
Time now is 11.448431581426.

        via
 D1 l   0   2
----+--------
   0l   1   4
   2l   3   1
   3l   8   3
Entity0 Update Complete. Distance Table is:
Time now is 11.812372776751488.

          via
 D0 l   1   2   3
----+------------
   1l   1   3 999
   2l   2   2   6
   3l   4   4   4
Entity3 Update Complete. Distance Table is:
Time now is 13.090669881792838.
        via
 D3 l   0   2
----+--------
   0l   7   5
   1l   6   3
   2l   7   2
Entity3 Update Complete. Distance Table is:
Time now is 14.681381220790833.
        via
 D3 l   0   2
----+--------
   0l   7   4
   1l   6   3
   2l   7   2
Entity1 Update Complete. Distance Table is:
Time now is 14.687493080188702.

        via
 D1 l   0   2
```

```
----+--------
   0|   1   3
   2|   3   1
   3|   8   3
Entity1 Update Complete. Distance Table is:
Time now is 15.854753681740853.

        via
 D1 |   0   2
----+--------
   0|   1   3
   2|   3   1
   3|   5   3
Entity0 Update Complete. Distance Table is:
Time now is 16.681338857005045.

          via
 D0 |   1   2   3
----+------------
   1|   1   3  12
   2|   2   2   6
   3|   4   4   4
Entity2 Update Complete. Distance Table is:
Time now is 16.791995097937466.

          via
 D2 |   0   1   3
----+------------
   0|   3   2   9
   1|   3   1  10
   3|   6   4   2
Entity0 Update Complete. Distance Table is:
Time now is 21.21464430905081.

          via
 D0 |   1   2   3
----+------------
   1|   1   3   7
   2|   2   2   6
   3|   4   4   4
Entity2 Update Complete. Distance Table is:
Time now is 22.059509862129246.

          via
 D2 |   0   1   3
----+------------
   0|   3   2   7
   1|   3   1   5
   3|   6   4   2
Entity0 Update Complete. Distance Table is:
Time now is 26.54603813462709.

          via
 D0 |   1   2   3
```

```
----+------------
   1l   1   3   7
   2l   2   2   6
   3l   4   4   4
Entity2 Update Complete. Distance Table is:
Time now is 30.99332519269049.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   6
   1l   3   1   5
   3l   6   4   2
Entity0 linkCostChange Handled. Distance Table is:
Time now is 10000.0.

           via
 D0 l   1   2   3
----+------------
   1l  20 999 999
   2l 999   2 999
   3l 999 999   7
Entity1 linkCostChange Handled. Distance Table is:
Time now is 10000.0.

         via
 D1 l   0   2
----+--------
   0l  20 999
   2l 999   1
   3l 999 999
Entity3 Update Complete. Distance Table is:
Time now is 10002.490030061554.
         via
 D3 l   0   2
----+--------
   0l   7   4
   1l   6   3
   2l   6   2
Entity0 Update Complete. Distance Table is:
Time now is 10003.197953875051.

           via
 D0 l   1   2   3
----+------------
   1l  20 999 999
   2l  21   2 999
   3l 999 999   7
Entity1 Update Complete. Distance Table is:
Time now is 10006.302956167443.

         via
 D1 l   0   2
----+--------
```

```
   0l  20 999
   2l  22   1
   3l  27 999
Entity2 Update Complete. Distance Table is:
Time now is 10006.398589523318.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   6
   1l   3   1   5
   3l   6   4   2
Entity1 Update Complete. Distance Table is:
Time now is 10007.413302965828.

         via
 D1 l   0   2
----+--------
   0l  20 999
   2l  22   1
   3l  27 999
Entity3 Update Complete. Distance Table is:
Time now is 10008.284436019267.
         via
 D3 l   0   2
----+--------
   0l   7   4
   1l   6   3
   2l   6   2
Entity2 Update Complete. Distance Table is:
Time now is 10008.565710858536.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   6
   1l   3   1   5
   3l   6   4   2
Entity0 Update Complete. Distance Table is:
Time now is 10009.521119089248.

           via
 D0 l   1   2   3
----+------------
   1l  20 999 999
   2l  21   2 999
   3l  47 999   7
Entity2 Update Complete. Distance Table is:
Time now is 10012.497817798245.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   6
   1l   3   1   5
   3l   6   4   2
Entity2 Update Complete. Distance Table is:
Time now is 10014.160658972132.

           via
 D2 l   0   1   3
----+------------
   0l   3   2   6
   1l   3   1   5
   3l   6   4   2
Simulator terminated at t=10014.160658972132, no packets in medium.
```