

ITCS 6166 – Computer Communications and Networks

Project 3- Implementing Distance Vector Routing Protocol

Shanmathi Rajasekar

800966697

Distance Vector Routing Protocol:

Routers using distance-vector protocol have no knowledge about the entire path from source to destination. Distance-vector protocols are based on calculating the minimum direction and distance to any link in a network, which is the destination.

It determines the next hop or next router which is nearest to the first. It measures the costs between two consecutive nodes. The least cost is the minimum distance between two nodes.

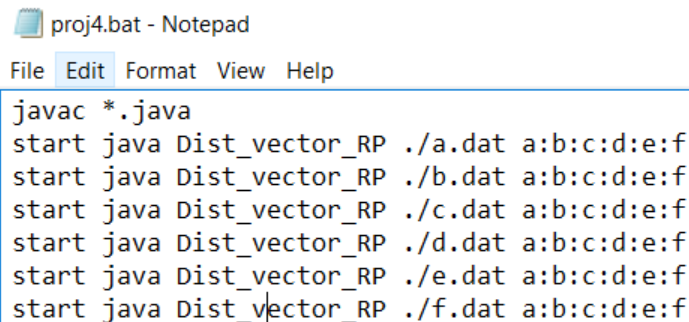
Each node maintains a vector of minimum distance to every node. The values in this can be changed dynamically. The input files are sensed every 15 seconds. If values are changed, the router tables get updated.

Steps to execute the code:

1. Download the CCN zip file and unzip it.
2. Edit inputs (to change costs) if required (the default is given costs and nodes)
3. Open command prompt and change your path ("CCN" should be the last folder)
4. To compile – javac Dist_vector_RP.java
5. To run - proj4.bat

```
C:\Users\shanm\Desktop\CCN>javac Dist_vector_RP.java  
  
C:\Users\shanm\Desktop\CCN>proj4.bat
```

6. If there is a path error, right click proj4.bat and click "edit"



```
proj4.bat - Notepad  
File Edit Format View Help  
javac *.java  
start java Dist_vector_RP ./a.dat a:b:c:d:e:f  
start java Dist_vector_RP ./b.dat a:b:c:d:e:f  
start java Dist_vector_RP ./c.dat a:b:c:d:e:f  
start java Dist_vector_RP ./d.dat a:b:c:d:e:f  
start java Dist_vector_RP ./e.dat a:b:c:d:e:f  
start java Dist_vector_RP ./f.dat a:b:c:d:e:f
```

Edit the input file path

Main (): The program starts with this main function. The other functions are called in this function.

Read (): This reads the values from the input file. It has data as required. This reads the connected nodes and costs to those nodes. It reads the inputs in correct format. The first line of the text file is a single number, which stands for the number of directly attached links.

Compute (): This method is implemented in the entity when the node 0 receives the packet from its neighbor and calculates the least distance from node 0 to every other node using Bellman ford equation through its neighbors.

Update (): It updates the DV table every 15 seconds. If any of the cost or neighbor is changed dynamically, it senses the change and updates the table. From next 'output number' the costs are computed according to the updated values.

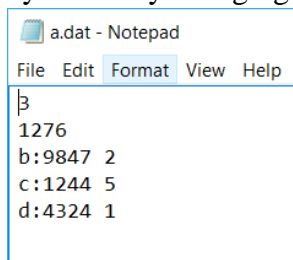
PrintV (): The printV () is called after each update which prints the updated the table. The entities are instantiated and the run () method calls the update () method internally to update the table and the printV() method is used to print the updated table.

The inputs are given in the input files – a.dat, b.dat, c.dat, d.dat, e.dat and f.dat. It includes number of nodes directly connected to it in first line. The neighboring nodes with its costs. Make sure the costs between 2 nodes are changed in both the input files. For eg., if cost between a and b is 2, make sure it is same in both a.dat and b.dat.

To check on recursive update, change the input of a input file dynamically and see for the changes in that terminal.

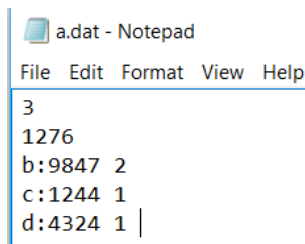
Input files:

Before dynamically changing the cost:



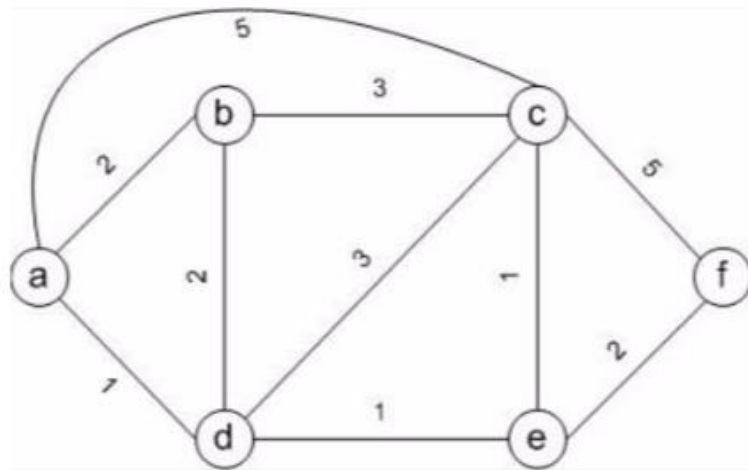
```
a.dat - Notepad
File Edit Format View Help
3
1276
b:9847 2
c:1244 5
d:4324 1
```

After dynamically changing the cost:



```
a.dat - Notepad
File Edit Format View Help
3
1276
b:9847 2
c:1244 1
d:4324 1 |
```

Given:



Compile and run:

```
C:\Users\shanm\Desktop\CCN>javac Dist_vector_RP.java
C:\Users\shanm\Desktop\CCN>proj4.bat
C:\Users\shanm\Desktop\CCN>javac *.java
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./a.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./b.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./c.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./d.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./e.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>start java Dist_vector_RP ./f.dat a:b:c:d:e:f
C:\Users\shanm\Desktop\CCN>
```

Outputs:

From a:

```
Output number 1
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is c and the cost is 5.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: No route
Shortest path a-f: No route
Output number 2
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 4.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is c and the cost is 10.0
Output number 3
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 3.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is d and the cost is 4.0
Output number 4
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 3.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is d and the cost is 4.0
```

From b:

```
Output number 1
Shortest path b-a: the next hop is a and the cost is 2.0
Shortest path b-c: the next hop is c and the cost is 3.0
Shortest path b-d: the next hop is d and the cost is 2.0
Shortest path b-e: No route
Shortest path b-f: No route
Output number 2
Shortest path b-a: the next hop is a and the cost is 2.0
Shortest path b-c: the next hop is c and the cost is 3.0
Shortest path b-d: the next hop is d and the cost is 2.0
Shortest path b-e: the next hop is d and the cost is 3.0
Shortest path b-f: the next hop is c and the cost is 8.0
Output number 3
Shortest path b-a: the next hop is a and the cost is 2.0
Shortest path b-c: the next hop is c and the cost is 3.0
Shortest path b-d: the next hop is d and the cost is 2.0
Shortest path b-e: the next hop is d and the cost is 3.0
Shortest path b-f: the next hop is d and the cost is 5.0
Output number 4
Shortest path b-a: the next hop is a and the cost is 2.0
Shortest path b-c: the next hop is c and the cost is 3.0
Shortest path b-d: the next hop is d and the cost is 2.0
Shortest path b-e: the next hop is d and the cost is 3.0
Shortest path b-f: the next hop is d and the cost is 5.0
```

From c:

```
Output number 1
Shortest path c-a: the next hop is a and the cost is 5.0
Shortest path c-b: the next hop is b and the cost is 3.0
Shortest path c-d: the next hop is d and the cost is 3.0
Shortest path c-e: the next hop is e and the cost is 1.0
Shortest path c-f: the next hop is f and the cost is 5.0
Output number 2
Shortest path c-a: the next hop is d and the cost is 4.0
Shortest path c-b: the next hop is b and the cost is 3.0
Shortest path c-d: the next hop is e and the cost is 2.0
Shortest path c-e: the next hop is e and the cost is 1.0
Shortest path c-f: the next hop is e and the cost is 3.0
Output number 3
Shortest path c-a: the next hop is e and the cost is 3.0
Shortest path c-b: the next hop is b and the cost is 3.0
Shortest path c-d: the next hop is e and the cost is 2.0
Shortest path c-e: the next hop is e and the cost is 1.0
Shortest path c-f: the next hop is e and the cost is 3.0
Output number 4
Shortest path c-a: the next hop is e and the cost is 3.0
Shortest path c-b: the next hop is b and the cost is 3.0
Shortest path c-d: the next hop is e and the cost is 2.0
Shortest path c-e: the next hop is e and the cost is 1.0
Shortest path c-f: the next hop is e and the cost is 3.0
```

From d:

```
Output number 1
Shortest path d-a: the next hop is a and the cost is 1.0
Shortest path d-b: the next hop is b and the cost is 2.0
Shortest path d-c: the next hop is c and the cost is 3.0
Shortest path d-e: the next hop is e and the cost is 1.0
Shortest path d-f: No route
Output number 2
Shortest path d-a: the next hop is a and the cost is 1.0
Shortest path d-b: the next hop is b and the cost is 2.0
Shortest path d-c: the next hop is e and the cost is 2.0
Shortest path d-e: the next hop is e and the cost is 1.0
Shortest path d-f: the next hop is e and the cost is 3.0
Output number 3
Shortest path d-a: the next hop is a and the cost is 1.0
Shortest path d-b: the next hop is b and the cost is 2.0
Shortest path d-c: the next hop is e and the cost is 2.0
Shortest path d-e: the next hop is e and the cost is 1.0
Shortest path d-f: the next hop is e and the cost is 3.0
Output number 4
Shortest path d-a: the next hop is a and the cost is 1.0
Shortest path d-b: the next hop is b and the cost is 2.0
Shortest path d-c: the next hop is e and the cost is 2.0
Shortest path d-e: the next hop is e and the cost is 1.0
Shortest path d-f: the next hop is e and the cost is 3.0
```

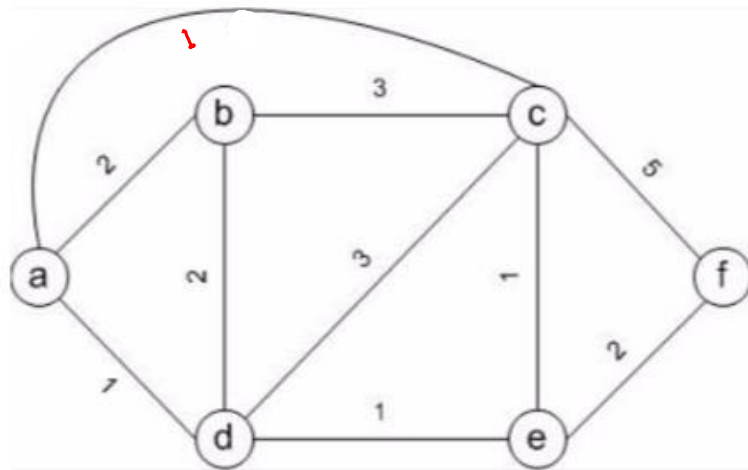
From e:

```
Output number 1
Shortest path e-a: No route
Shortest path e-b: No route
Shortest path e-c: the next hop is c and the cost is 1.0
Shortest path e-d: the next hop is d and the cost is 1.0
Shortest path e-f: the next hop is f and the cost is 2.0
Output number 2
Shortest path e-a: the next hop is d and the cost is 2.0
Shortest path e-b: the next hop is d and the cost is 3.0
Shortest path e-c: the next hop is c and the cost is 1.0
Shortest path e-d: the next hop is d and the cost is 1.0
Shortest path e-f: the next hop is f and the cost is 2.0
Output number 3
Shortest path e-a: the next hop is d and the cost is 2.0
Shortest path e-b: the next hop is d and the cost is 3.0
Shortest path e-c: the next hop is c and the cost is 1.0
Shortest path e-d: the next hop is d and the cost is 1.0
Shortest path e-f: the next hop is f and the cost is 2.0
Output number 4
Shortest path e-a: the next hop is d and the cost is 2.0
Shortest path e-b: the next hop is d and the cost is 3.0
Shortest path e-c: the next hop is c and the cost is 1.0
Shortest path e-d: the next hop is d and the cost is 1.0
Shortest path e-f: the next hop is f and the cost is 2.0
```

From f:

```
Output number 1
Shortest path f-a: No route
Shortest path f-b: No route
Shortest path f-c: the next hop is c and the cost is 5.0
Shortest path f-d: No route
Shortest path f-e: the next hop is e and the cost is 2.0
Output number 2
Shortest path f-a: the next hop is c and the cost is 10.0
Shortest path f-b: the next hop is c and the cost is 8.0
Shortest path f-c: the next hop is e and the cost is 3.0
Shortest path f-d: the next hop is e and the cost is 3.0
Shortest path f-e: the next hop is e and the cost is 2.0
Output number 3
Shortest path f-a: the next hop is e and the cost is 4.0
Shortest path f-b: the next hop is e and the cost is 5.0
Shortest path f-c: the next hop is e and the cost is 3.0
Shortest path f-d: the next hop is e and the cost is 3.0
Shortest path f-e: the next hop is e and the cost is 2.0
Output number 4
Shortest path f-a: the next hop is e and the cost is 4.0
Shortest path f-b: the next hop is e and the cost is 5.0
Shortest path f-c: the next hop is e and the cost is 3.0
Shortest path f-d: the next hop is e and the cost is 3.0
Shortest path f-e: the next hop is e and the cost is 2.0
```

When the cost of a-c is changed from 5 to 1 in a.dat, dynamically:



From a:

```

Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 3.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is d and the cost is 4.0
Output number 6
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 3.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is d and the cost is 4.0
Output number 7
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is d and the cost is 3.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is d and the cost is 2.0
Shortest path a-f: the next hop is d and the cost is 4.0
Output number 8
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is c and the cost is 1.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is c and the cost is 2.0
Shortest path a-f: the next hop is c and the cost is 4.0
Output number 9
Shortest path a-b: the next hop is b and the cost is 2.0
Shortest path a-c: the next hop is c and the cost is 1.0
Shortest path a-d: the next hop is d and the cost is 1.0
Shortest path a-e: the next hop is c and the cost is 2.0
Shortest path a-f: the next hop is c and the cost is 4.0

```

Till Output number= 7, the total least cost from a to c is 3 (a:d:e:c), in a.dat file when the a-c cost is changed to 1 dynamically, in the next update(every 15 sec), it gets updated. Hence at output number-8, the a-c cost is 1(min value).