Computer Communications & Networks

Febin Zachariah – 800961027

Aravind Prasannakumari Vijayakumar -800896112

Project-3

Implementation of Distance Vector Routing Protocol

We have implemented distance vector routing protocol using java programming language. We have created two java files: 1) Admin.Java and 2) MainRouter.Java. We are using singe machine for implementing this project.

Admin will access the given directory path to find the number of routers present and it will assign port numbers corresponding to each router/file. For each router present, **MainRouter** java class is called and it calculates distance to other routers and it detects link cost changes and recalculate the cost.

Work Flow of Admin and MainRouter

1) Admin will accept input from user (Directory Path).
2) It checks whether the Directory path is valid or not.
3) Calculate how many files are present in the directory (Each file corresponds to router.)
4) Asking the user to assign port number for each file present in the directory.
5) Start processes/threads for each file in the directory by using Process builder in java and using arguments (Router id, total number of routers, directory path, Router Name with corresponding Port number).

```
ProcessBuilder processBuilder = new ProcessBuilder("cmd.exe", "/c", "start java MainRouter " + (i + 1) + " \""
        + data[i].getParent().replace("\\", "/") + "\" " + size + allNodes);
try {
```

6) Now the control is passed to MainRouter class. Separate command prompt window is opened for each router.
7) Initialize the local variables like hop List, neighbor List, Socket and File.
8) Now we are creating two threads for each router.
9) One thread is used for reading the distance vector information from the network and update the global network distance matrix of each router.
10) Another thread is opened to calculate the shortest path based on the distance vector algorithm and write the updated information to neighbors.
11) After each writing is completed, it waits for 6 seconds to receive distance vector information from its neighbors.
12) Then it performs computations and again it waits for 12 seconds and repeat the same process from step 9.

Running Instructions

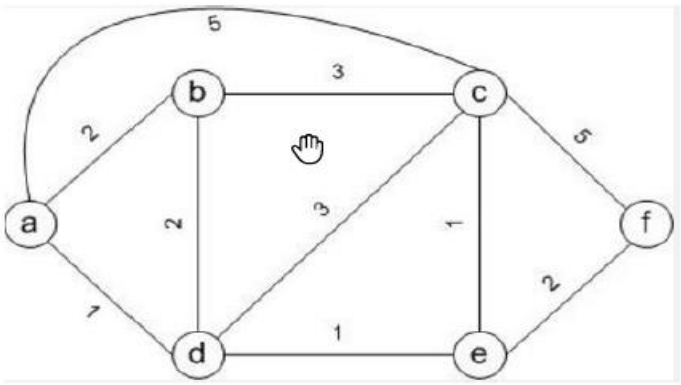
1) Got to the Source_Code folder and compile the files

   javac *.java

2) Run Admin.java file and give the corresponding directory path as argument

   java Admin *C:\Users\febin\Desktop\Project3\Data_6*

3) We are executing distance vector algorithm for the following network topology.



4) Assign port number for each file. Error handling is done to avoid port number conflict and incorrect port number.

```
C:\Users\febin\Desktop\Project3\Source_Code>java Admin C:\Users\febin\Desktop\Project3\Data_6
Initilization of Port Number to 6 Routers
Enter Port No: for Router: a
1234
Enter Port No: for Router: b
1234
Address is Already in Use:
2345
Enter Port No: for Router: c
4567
Enter Port No: for Router: d
4321
Enter Port No: for Router: e
12345678
Enter a valid Port Number > 1024 && < 65536
7890
Enter Port No: for Router: f
1111
DV Algorithm Started
```

5) Initial terminal output for each router is given below:

```
> 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 found
shortest path a-f:  no route found
> output number 2
```

```
> 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 found
shortest path b-f:  no route found
```

```
> 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 1

hortest path d-a : the next hop is a and the cost is 1.0
hortest path d-b : the next hop is b and the cost is 2.0
hortest path d-c : the next hop is c and the cost is 3.0
hortest path d-e : the next hop is e and the cost is 1.0
hortest path d-f:  no route found
output number 2
```

```
> output number 1

shortest path e-a:  no route found
shortest path e-b:  no route found
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
```

```
> output number 1

shortest path f-a:  no route found
shortest path f-b:  no route found
shortest path f-c : the next hop is c and the cost is 5.0
shortest path f-d:  no route found
shortest path f-e : the next hop is e and the cost is 2.0
```

6) Shortest path for each Router

```
> 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
```

```
> output number 5

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 5

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 6
```

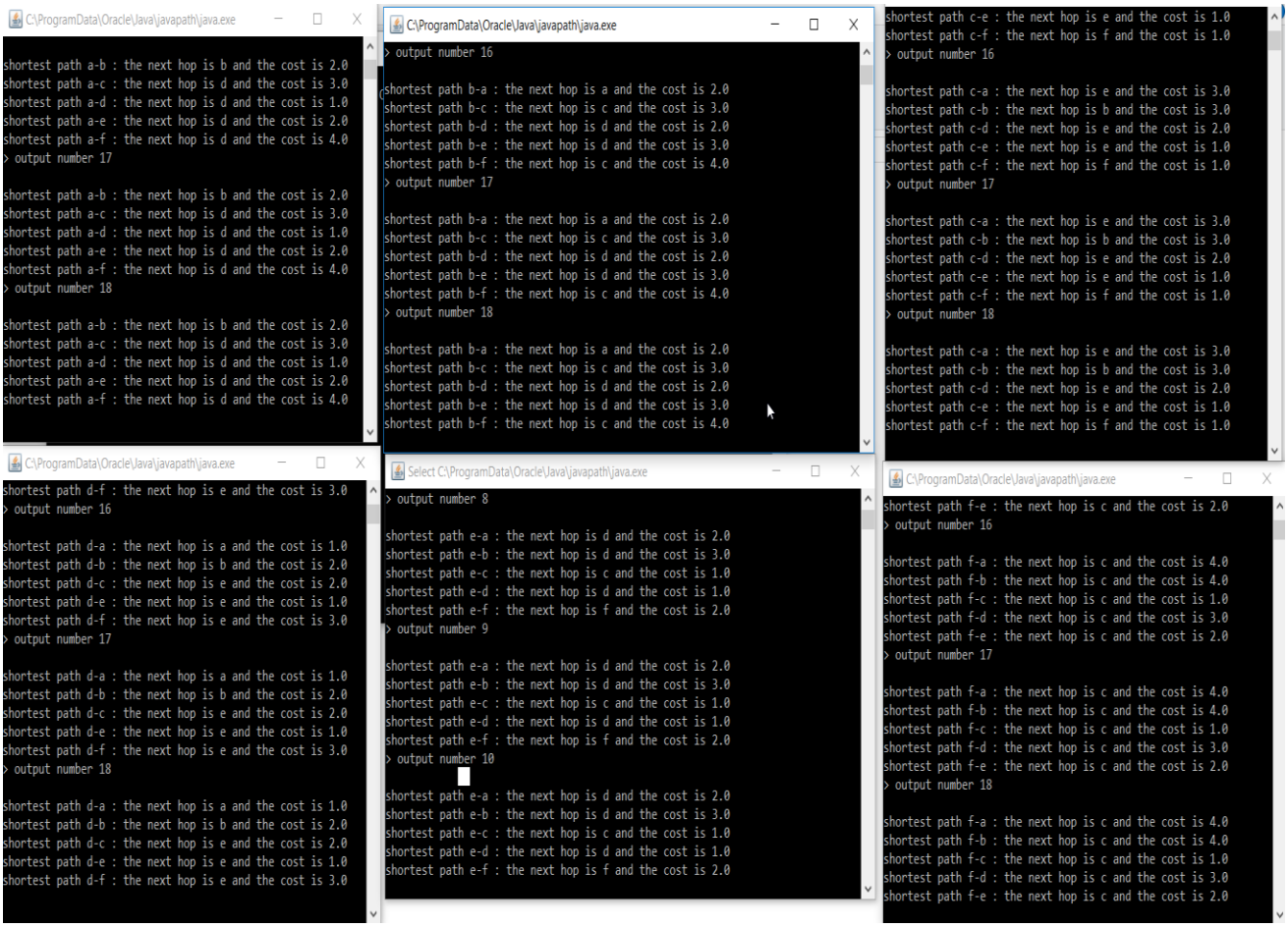
```
shortest path d-a : the next hop is a and the cost is 1.0
shortest path d-b : the next hop is b and the cost is 2.0
shortest path d-c : the next hop is e and the cost is 2.0
shortest path d-e : the next hop is e and the cost is 1.0
shortest path d-f : the next hop is e and the cost is 3.0
> output number 8
```

```
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 9
```

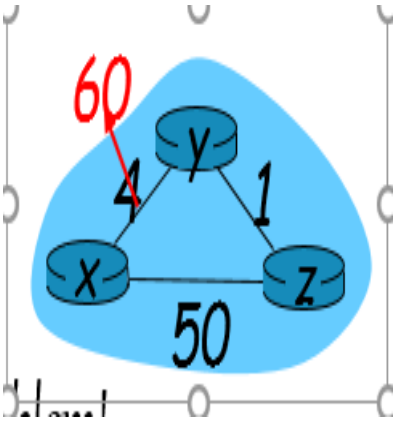
```
shortest path f-a : the next hop is e and the cost is 4.0
shortest path f-b : the next hop is e and the cost is 5.0
shortest path f-c : the next hop is e and the cost is 3.0
shortest path f-d : the next hop is e and the cost is 3.0
shortest path f-e : the next hop is e and the cost is 2.0
> output number 10

shortest path f-a : the next hop is e and the cost is 4.0
```

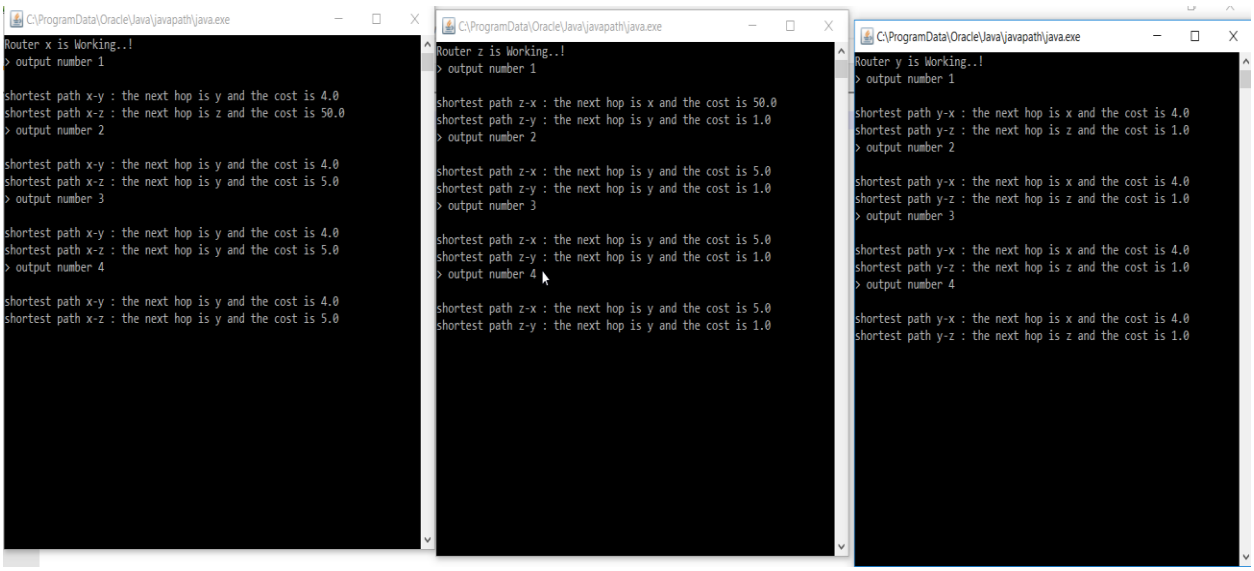
7) Link Cost Change Handling

Changing the link cost of C-F from 5 to 1



8) Similarly, Recursive Update Problem is also handled. Consider the following network topology.





After link cost changes to large value