ITCS 6166: Project 3 – Distance Vector Routing Protocol

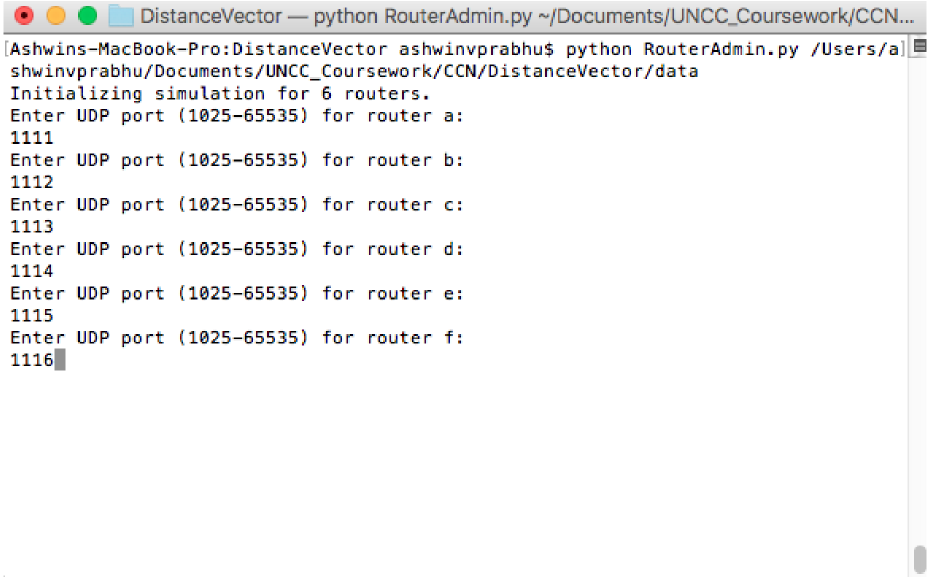
Submitted by,
Ashwin Venkatesh Prabhu (800960400)

Distance Vector Routing Protocol and Implementation details

1) Distance vector routing is a simple protocol used in packet switching networks that utilizes distance to decide the best packet forwarding path. Distance is typically represented by hop count.
2) They are simple and require little management, and are very efficient for small networks. But they do not scale well when it comes to large network.
3) This project is implemented using Python 3.5.
4) The program is running on a single machine (a simulation of the protocol). A router in represented by a terminal window (If there are five routers, then five terminal windows will pop up)
5) Have used an external library called "terminal.py" to dynamically generate a terminal window for the router, as and when the input is read.
   (Ref: https://github.com/skywind3000/terminal)
6) Input is a set of ".dat" files, each file consisting of routers attached links and their costs.
7) The program at each router does not know the complete network topology and only know about the routers which they are connected to.
8) Routing program at each router reports the cost and next hop for the shortest path to all other routers in the network.
9) Each router sends the routing information to all other connected routers at a certain frequency. This improves the robustness of the protocol.

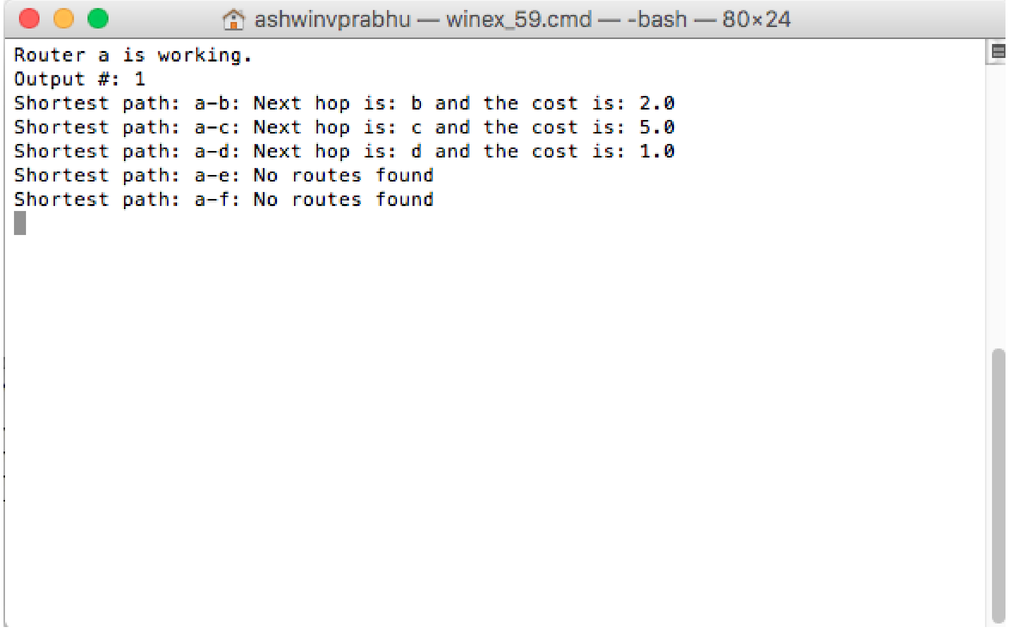
Results

1) The network topology used here is the same as the one given in the problem description.
2) Initial command to start the program is *"python RouterAdmin.py <input data folder path>"*
3) "RouterAdmin.py" will start n number of routers depending on the number of input files available.
4) After running the command mentioned in step 1, user must input the port information needed for all the routers. Please refer to the image shown below:


DistanceVector — python RouterAdmin.py ~/Documents/UNCC_Coursework/CCN...

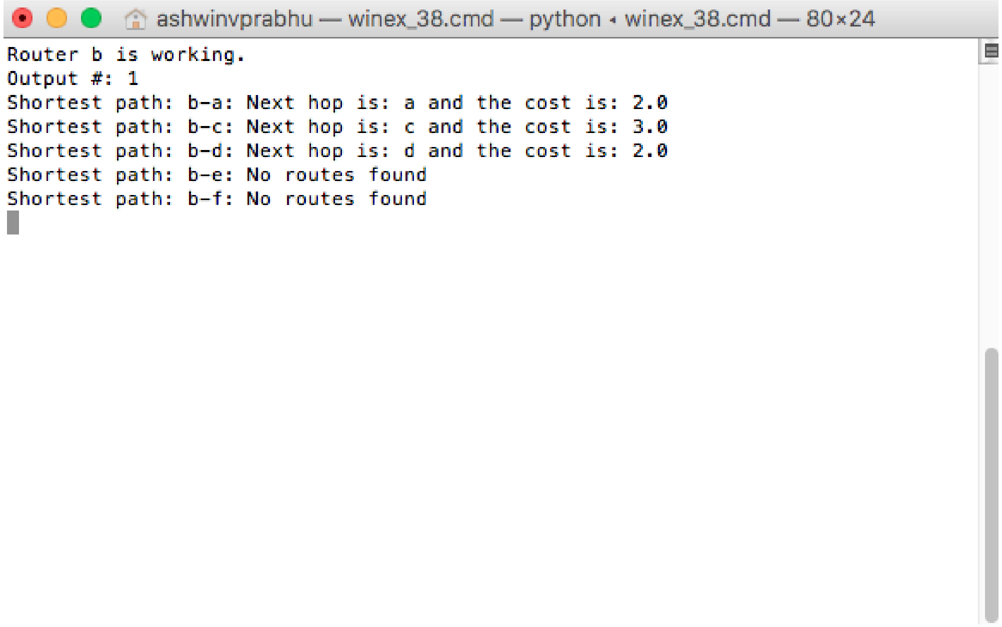
```
[Ashwins-MacBook-Pro:DistanceVector ashwinvprabhu$ python RouterAdmin.py /Users/a]
shwinvprabhu/Documents/UNCC_Coursework/CCN/DistanceVector/data
Initializing simulation for 6 routers.
Enter UDP port (1025-65535) for router a:
1111
Enter UDP port (1025-65535) for router b:
1112
Enter UDP port (1025-65535) for router c:
1113
Enter UDP port (1025-65535) for router d:
1114
Enter UDP port (1025-65535) for router e:
1115
Enter UDP port (1025-65535) for router f:
1116
```

5) After entering the port info for all the routers, the routers will be initialized and will start sending the routing table to all neighbors. Please refer to the screenshots below:
For router A:



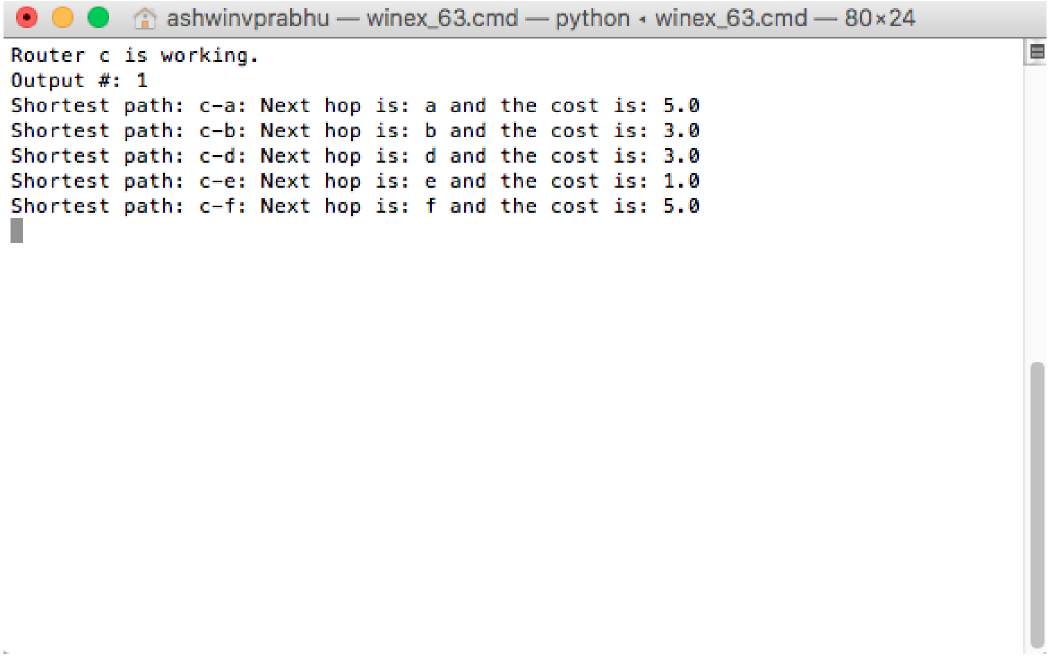
```
Router a is working.
Output #: 1
Shortest path: a-b: Next hop is: b and the cost is: 2.0
Shortest path: a-c: Next hop is: c and the cost is: 5.0
Shortest path: a-d: Next hop is: d and the cost is: 1.0
Shortest path: a-e: No routes found
Shortest path: a-f: No routes found
```

For router B:


ashwinvprabhu — winex_38.cmd — python ◂ winex_38.cmd — 80×24

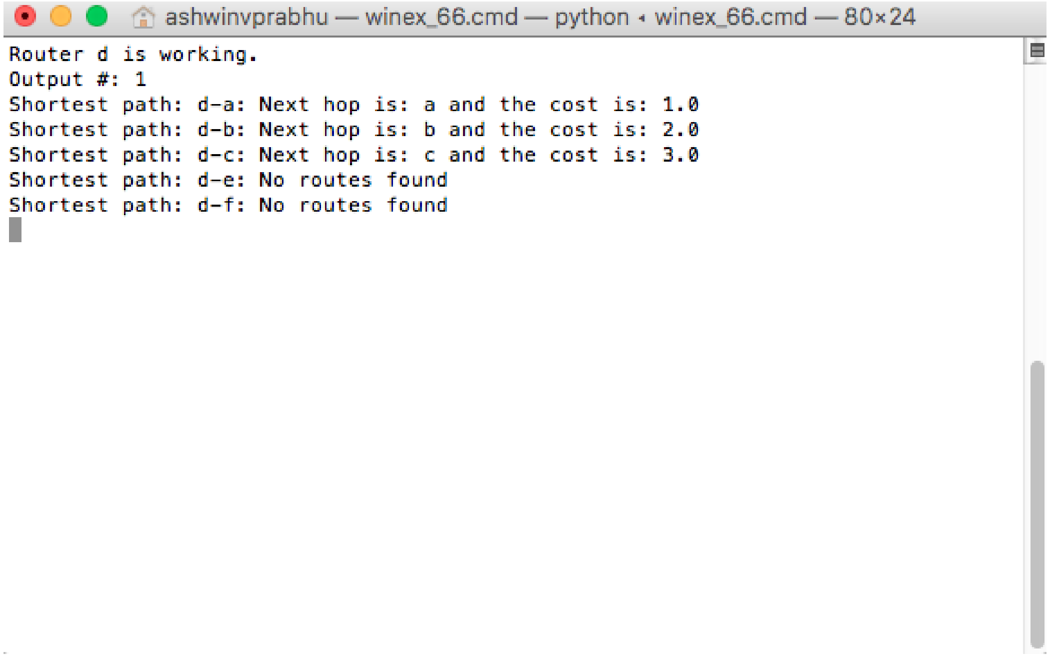
```
Router b is working.
Output #: 1
Shortest path: b-a: Next hop is: a and the cost is: 2.0
Shortest path: b-c: Next hop is: c and the cost is: 3.0
Shortest path: b-d: Next hop is: d and the cost is: 2.0
Shortest path: b-e: No routes found
Shortest path: b-f: No routes found
```

For router C:


ashwinvprabhu — winex_63.cmd — python ◂ winex_63.cmd — 80×24

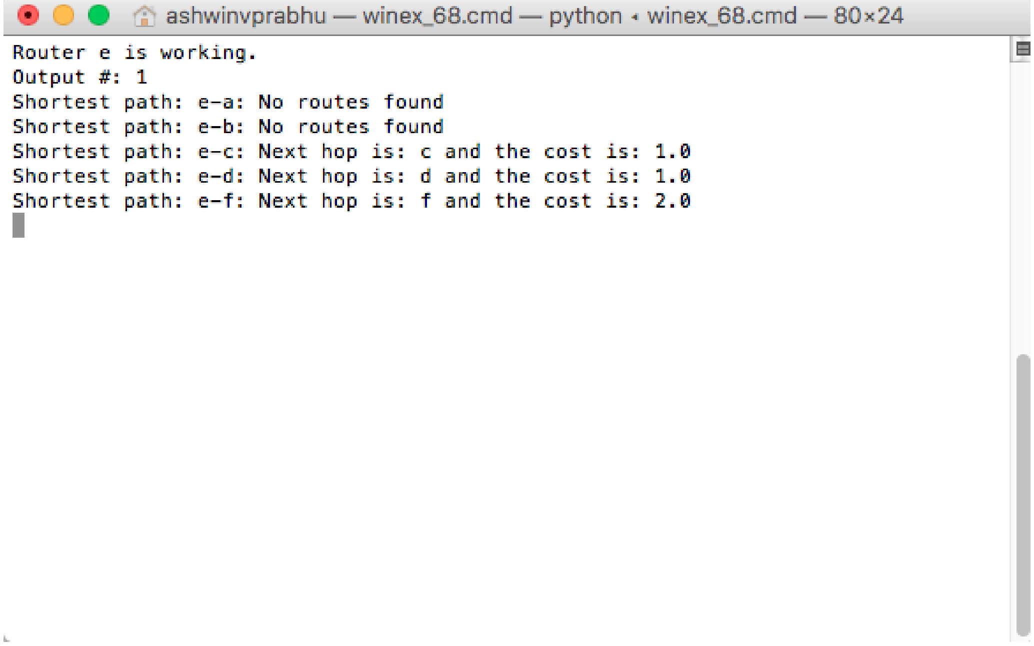
```
Router c is working.
Output #: 1
Shortest path: c-a: Next hop is: a and the cost is: 5.0
Shortest path: c-b: Next hop is: b and the cost is: 3.0
Shortest path: c-d: Next hop is: d and the cost is: 3.0
Shortest path: c-e: Next hop is: e and the cost is: 1.0
Shortest path: c-f: Next hop is: f and the cost is: 5.0
```

For router D:


ashwinvprabhu — winex_66.cmd — python ◂ winex_66.cmd — 80×24

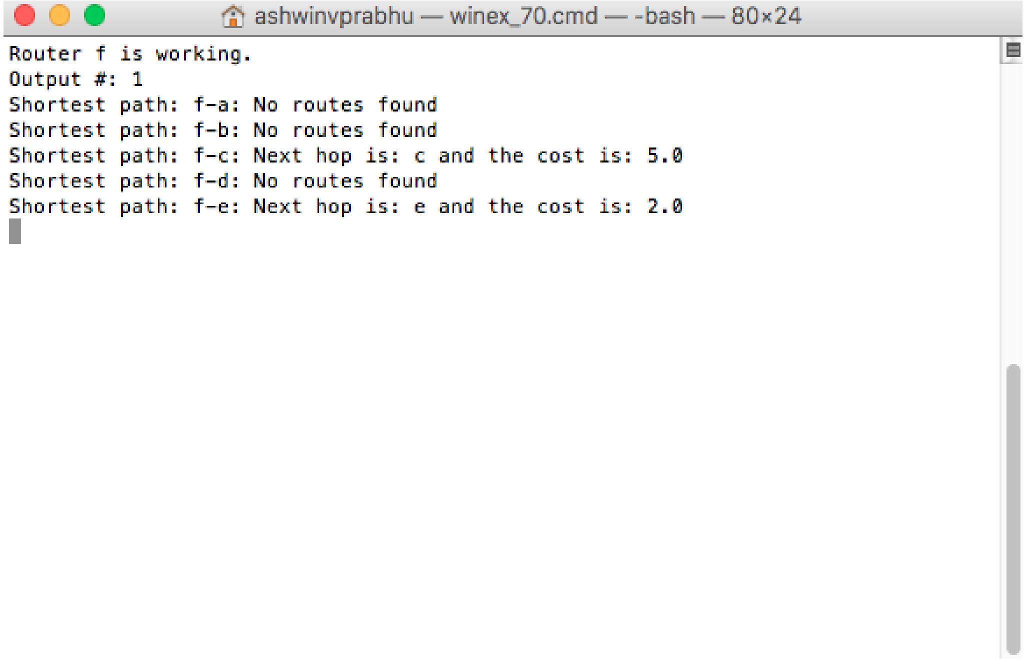
```
Router d is working.
Output #: 1
Shortest path: d-a: Next hop is: a and the cost is: 1.0
Shortest path: d-b: Next hop is: b and the cost is: 2.0
Shortest path: d-c: Next hop is: c and the cost is: 3.0
Shortest path: d-e: No routes found
Shortest path: d-f: No routes found
```

For router E:


ashwinvprabhu — winex_68.cmd — python ◂ winex_68.cmd — 80×24

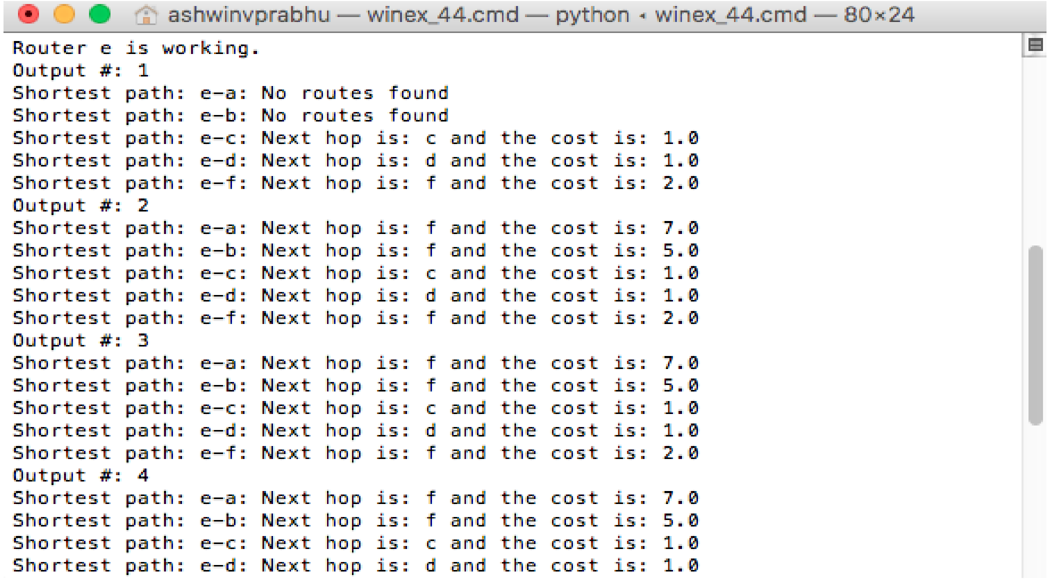
```
Router e is working.
Output #: 1
Shortest path: e-a: No routes found
Shortest path: e-b: No routes found
Shortest path: e-c: Next hop is: c and the cost is: 1.0
Shortest path: e-d: Next hop is: d and the cost is: 1.0
Shortest path: e-f: Next hop is: f and the cost is: 2.0
```

For router F:



```
Router f is working.
Output #: 1
Shortest path: f-a: No routes found
Shortest path: f-b: No routes found
Shortest path: f-c: Next hop is: c and the cost is: 5.0
Shortest path: f-d: No routes found
Shortest path: f-e: Next hop is: e and the cost is: 2.0
```

For example, after a couple of hops, can see the routing table updates at router E as follows:


ashwinvprabhu — winex_44.cmd — python ◂ winex_44.cmd — 80×24

```
Router e is working.
Output #: 1
Shortest path: e-a: No routes found
Shortest path: e-b: No routes found
Shortest path: e-c: Next hop is: c and the cost is: 1.0
Shortest path: e-d: Next hop is: d and the cost is: 1.0
Shortest path: e-f: Next hop is: f and the cost is: 2.0
Output #: 2
Shortest path: e-a: Next hop is: f and the cost is: 7.0
Shortest path: e-b: Next hop is: f and the cost is: 5.0
Shortest path: e-c: Next hop is: c and the cost is: 1.0
Shortest path: e-d: Next hop is: d and the cost is: 1.0
Shortest path: e-f: Next hop is: f and the cost is: 2.0
Output #: 3
Shortest path: e-a: Next hop is: f and the cost is: 7.0
Shortest path: e-b: Next hop is: f and the cost is: 5.0
Shortest path: e-c: Next hop is: c and the cost is: 1.0
Shortest path: e-d: Next hop is: d and the cost is: 1.0
Shortest path: e-f: Next hop is: f and the cost is: 2.0
Output #: 4
Shortest path: e-a: Next hop is: f and the cost is: 7.0
Shortest path: e-b: Next hop is: f and the cost is: 5.0
Shortest path: e-c: Next hop is: c and the cost is: 1.0
Shortest path: e-d: Next hop is: d and the cost is: 1.0
```

6) Link cost change feature is not implemented here.

References

1) https://github.com/skywind3000/terminal