National University of Sciences & Technology
School of Electrical Engineering and Computer Science
Department of Computing
EE353: Computer Networks, BESE-6 Spring 2017

Project Specifications (V1)	
CLO 4: Design and implement solutions to contemporary networking issues (through hands on programming)	
Maximum Marks: 10	Instructor: Dr. Arsalan Ahmad
Date: 17th April 2017	Due Date: mid night 22nd May 2017

1 General:

This is a group assignment. Maximum size of the group is restricted to two members.

Please avoid plagiarism; any such case would result in award of zero marks both to the "sharer" and the "acquirer". Maximum score is 20 points that would be scaled back to 10 marks.

2 Learning Objectives

In this assignment your task is to implement the distance-vector routing protocol. Your program will be running at all routers in the specified network. At each router the input to your program is a set of directly attached routers (i.e. neighbors) and the costs of these links. Each router learns routes from its neighboring routers' perspectives and then advertises the routes from its own perspective. Your implementation at each router should report the least cost path and the associated cost to all other routers in the network. Your program should be able to deal with changing route costs as well.

On completing this assignment you will gain sufficient expertise in the following skills:

a) Designing a routing protocol
b) Distance-Vector (Bellman-Ford) routing algorithm
c) Socket programming in Python
d) Multi-threading in Python
e) Handling routing dynamics

3 Specifications:

In this assignment, you will implement the distance-vector routing protocol using Python as the programming language. You are provided with the topology map in the form of configuration files. The program should be named DVR.py and it will accept the following command line arguments:

DVR.py <router-id> <port-no> <router-config-file>

For example:

DVR.py A 5000 ConfigA.txt

The router configuration file, for example ConfigA.txt, is the configuration file for Router A that has the following details:

```
2
B 6.5 5001
F 2.2 5005
```

The first line of this file indicates the number of neighbors for Router A. Note that it is not the total number of routers in the network. Following this, there is one line dedicated to each neighbor. It starts with the neighbor ID, followed by the cost to reach this neighbor and finally the port number that this neighbor is using for listening. For example, the second line in the configA.txt above indicates that the cost to neighbor B is 6.5 and this neighbor is using port number 5001 for receiving distance-vector packets. The router ids will be uppercase single letter alphabets. The link costs should be floating point numbers (up to the first decimal) and the port numbers should be integers. These three fields will be separated by a single white space between two successive fields in each line of the configuration file. Further, the link costs will be consistent in both directions, i.e., if the cost from A to B is 6.5, then the link from B to A will also have a cost of 6.5.

Each router must only know the costs to reach its direct neighbors and the cost it takes to reach other routers from any given neighbor. The routers must not have global knowledge (i.e. Information about the entire network topology). Upon initialization, each router creates a distance-vector update packet (containing the appropriate information – see description of distance-vector routing protocol in the textbook) and sends this packet to all direct neighbors. The exact format of the distance-vector packets that you will use is left for you to decide. Upon receiving this distance-vector update packet, each neighboring router will incorporate the provided information into its routing table. Each router should periodically broadcast the distance-vector update packet to its neighbors every 10 seconds.

You are required to use **UDP** as the transport protocol for exchanging link-state packets amongst the neighbors.

On receiving distance-vector update packets from all other routers, a router can build up a **reachability matrix**. Given a view of the neighboring routers and their reachability, a router should run the Bellman-Ford algorithm to compute least-cost paths to all other routers within the network. Each router should wait for 60 seconds since start-up and then execute Bellman-Ford.

Once a router finishes running Bellman-Ford algorithm, it should print out to the terminal, the least cost path to each destination router in the topology (excluding itself) along with the cost of this path. The following is an example output for router A in some arbitrary network:

> I am Router A
> Least cost path to router B: ACB and the cost: 14.2
> Least cost path to router C: AC and the cost: 2.5

Your program should execute forever (as a loop). In other words, each router should keep broadcasting distance-vector value packets every 10 seconds and Bellman-Ford algorithm should be executed every time a change happens.

3.2 Dealing with changes in link cost:

You must ensure that your algorithm is robust to changes in link costs and failures. Once the cost of link changes the network must be able to recon verge to accommodate the costs. The routing protocol should appropriately handle “good news” (i.e. shorter path becomes available) and “bad news” (i.e. shortest path no longer shortest).

You must provide an interface which can be used to display every link-cost in a network and give the ability to modify any of them. A simple method would be to run a separate thread that can read and write to the link cost values.

Once the cost of a link changes, the connected routers must recalculate the cost of reaching other nodes and must also provide an update to its neighbors, who will then notify their neighbors and so on until the network converges.

Recall that each router executes Bellman-Ford only when a link-cost changes. While reconverging, it may also happen that the distance values a router uses involve the link whose cost just changed. This may lead to the Count To Infinity problem. Handle this in your code using a method like Poison Reverse or Split Horizon.

3.3 Multi-threading

You must use multi-threading in your implementation. We recommend that you use at least three separate threads for listening (for receiving distance-vector updates), sending (for sending distance-vector updates after every 1 sec) and Bellman-Ford calculations (when an link-cost changes). You may choose to use more than three threads as per your requirement.

3.4 Simulated network

Since, we do not have access to real network routers (for implementation and testing of your programs), we will use a simulated network that will run on a single desktop machine. You will run different instances of your program on the same machine (use 'localhost'), however each instance of the routing protocol (corresponding to each router in the network) will be listening on a different port number. You can open different terminal windows (one each for every router in the network topology) to observe the behavior of your implementation. The topology size is restricted to maximum of 10 routers.

If your routing implementation executes correctly on a single desktop machine, it should also work correctly on real network routers.

3.5 Test Topology

You have been provided with configuration files for a sample topology of 6 routers. Use this topology to incrementally test your implementation. The correct output for Router A before and after failure of Router D is given. You should check your implementation for correctness at all routers with multiple router failures. Your implementation would be graded with a "Mystery" topology of maximum 10 routers with multiple router failures.

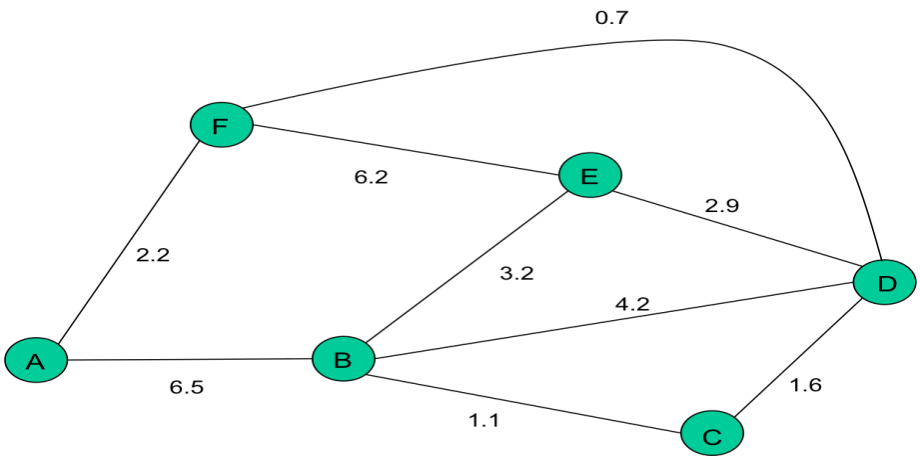

Fig: 1 Sample 6 Routers topology

Output with all routers working:

I am Router A
Least cost path to router C: AFDC and the cost: 4.5
Least cost path to router B: AFDCB and the cost: 5.6
Least cost path to router E: AFDE and the cost: 5.8
Least cost path to router D: AFD and the cost: 2.9
Least cost path to router F: AF and the cost: 2.2

Output after Router D fails:

I am Router A
Least cost path to router C: ABC and the cost: 7.6
Least cost path to router B: AB and the cost: 6.5
Least cost path to router E: AFE and the cost: 8.4
Least cost path to router F: AF and the cost: 2.2

3.5 Submission & Report:

You are required to submit your source code and a short report to an upload link that would be made available on the LMS. Zip your source code files and your report document in a file named Project_Surname_Surname (Surname of both Group members). Nominate one of the group members to submit on behalf of the group. Please note that you must upload your submission BEFORE the deadline. The LMS would continue accepting submissions after the due date. Late submissions would carry penalty per day with maximum of 2 days late submission allowed (see Section 3.6). Students who fail to submit would not be allowed to appear in viva and those who miss the viva would not be allowed to retake the viva.

Your submitted code would be checked for similarities and any instances of plagiarism would result in award of ZERO marks for all such students, irrespective of who has shared with whom. No exceptions!!

You must write down group members' Registration No's and Names at the beginning of the report**!!** All reports will be read for marking purposes.

The size of your report **MUST be under 2 pages**. Your report should briefly document your techniques and methodology used for implementation and how you combat the relevant problems in development. Treat it as a summary document only (point form is acceptable). It should act a reference for your instructor to quickly figure out what you have and haven't completed, how you did it, and it should mention anything you think is special about your system. You will be asked to demonstrate your program during your viva.

3.6 Late Submission Penalty:

Late penalty (on your received marks) will be applied as follows:

1 day after deadline: 25% reduction

2 days after deadline: 50% reduction

3 days after deadline: Not accepted