Project 3: Distance Vector Routing (DV)

Instructor: Dr. Shengquan Wang

Due Time:
Part 1: 6PM, 11/12/2014
Part 2: 6PM, 11/26/2014

The purpose of this project is to learn how distributed dynamic routing protocols like distance vector accomplish routing in practice. In this assignment, you will be asked to build a distance vector routing protocol that implements the distributed Bellman-Ford algorithm.

1 Design

We will use the running processes to simulate routers. Processes can be run on different hosts. Each router is assigned with a unique router ID (English letter). Two processes communicate with another process through its UDP socket.

Each router will maintain a distance vector as follows:

```
A B  7 1 B
A C 10 2 B
```

There are two entries in this distance vector. For each entry, the five columns are source router ID, destination router ID, distance, number of hops, the router ID for the next router in routing, respectively. Every router will also maintain its neighboring routers' distance vectors.

Upon receiving any distance vector entries from any neighboring router, the router will use the `Bellman-Ford Equation` to update its distance vector. Whenever its distance vector is changed, it will broadcast the changed (only changed) entries to its neighboring routers through a UDP socket and print out the changed entries to the console. UDP sockets in all routers should alway be listening (for potential messages from some neighbors).

Each router will also periodically broadcast advertisement to its neighbor (e.g., every 10 s). If a router doesn't hear one of its neighbor's advertisement within a timeout interval (e.g., 30 s), it will remove this neighbor and update its distance vector and broadcast the changed entries to its neighboring routers.

Here is what are expected with the multiple running routers. We use the same executable file for all routers. Name it as `dvrouter`. For example, run Router A in the shell as

```
prompt> dvrouter A 2547
```

where 2547 is the assigned UDP port number. Once a router A starts, it shall display as shown in the following example:

```
Router A (localhost, 2547) is active. Waiting for input and output ...
```

where `(localhost, 2547)` is the pair of the hostname and the port number.

The following input commands will be supported in the console for a router (e.g., router A):

add It adds a new link to the router. For example

```
add B localhost 3721 5
```

It will add the link between A and B (with hostname "localhost" and port number 3721) with a link cost 5 to Router A. If the router B is alive, a message will be sent to B and it will add the link between B and A with a link cost 5 to Router B. On Router A, it will display "The link A-B is set!" message; on Router B, it will display "The link B-A is set!" message. ~~If the router B is not alive, it should display a "B is not alive!" message on Router A.~~ Keep in mind the link cost is bidirectionally symmetric.

change It changes an existing link to the router. For example

```
change B 7
```

It will change the link between A and B with a link cost 7 to Router A. If the router B is alive, a message will be sent to B and it will change the link cost between B and A with a link cost 5 in Router B. On Router A, it will display "The cost of link A-B is changed!" message; on Router B, it will display "The cost of link B-A is changed!" message. ~~If the router B is not alive, it should display a "B is not alive!" message on Router A.~~ Keep in mind the link cost is bidirectionally symmetric.

display It display the distance vector of the router. Input

```
display
```

then we have the output as

```
The distance vector for Router A is:
A B  7 1 B
A C 10 2 B
```

kill It kills the running router. Input

```
kill
```

then we have the output as

```
Router A is about to be terminated...
```

2 Testing

Start 5 processes (as 5 routers) in 5 different terminals (they will be run on the same host). Each router is given a router ID. Its running process is associated with its host name and the port number. We assume they are

```
Router A: (localhost, 2547)
Router B: (localhost, 3721)
Router C: (localhost, 4568)
Router D: (localhost, 5834)
Router E: (localhost, 8710)
```

Make sure they are active before adding any links. This 5 routers will form the following network topology:

```
A -------- B
|    1     |\ 6
|          | \
| 2      3 |  E
|          | /
|    3     |/ 2
C -------- D
```

We assume all links are symmetric in terms of link cost in both directions.

Perform the following actions:

- Add all links to the routers using the command `add`;
- Change the link cost of B-E to 1 using the command `change` at Router B;
- Change the link cost of A-B to 4 using the command `change` at Router A;
- Display the distance vector for each router using the command `display`.

Once the system is stabilized, B detects a link cost of BA from 1 to 60. Perform the following actions:

- Change the link cost of B-A to 60 using the command `change` at Router B;
- Print out the number of messages for all nodes before the system is stabilized again;
- Kill router A.

You are required to print out to the console on any message received by each router and any change of DV at each router.

Next, apply `Poison Reverse` technique in the design of the DV. Re-perform the above actions again and show the difference after applying the `Poison Reverse` technique.

3 Submission

There are two parts of submission:

Part 1 (20% **Credits)** Form a group if you want to work with another student. In this part, you are going to report the following things:

(a) Describe in details the system architecture. Use a digram to show the message exchange flow.

(b) Write down a tackle strategy for this project on how to implement each command.

(c) Divide the implementation into tasks and draw a time table to show your plan to implement each task. If you work as a group of two, specify the task assignment for each member.

It is like when you work on a project you need to sit down to think or discuss how to do it. It should be written in a ".docx", ".doc", or ".pdf" format, and submitted to Project P3-1 on Canvas.

Part 2 (80% **Credits)** In this part, you are going to report the following things:

(a) Describe in details how you implemented the DV.

(b) Write a detailed report with the screenshots of the testing results for the two scenarios: without `Poison Reverse` and with `Poison Reverse`. Each screenshot should include your username and the current time, which show that you did it by yourself.

(c) Specify the contribution made by each member if you work as a group.

The report should be written in a ".docx", ".doc", or ".pdf" format. All source codes should be well formatted with good comments. Submit the report and the source code files to Project P3-2 on Canvas *individually*. Any `.zip` or `.tar` format is not permitted.