CS 640 Introduction to Computer Networks **Spring 2009**

http://pages.cs.wisc.edu/s̃uman/courses/wiki/doku.php?id=640-spring2009

Programming Assignment 2: Routing Protocol

Assigned: March 3, 2009 *Due: March 24, 2009, 11:59pm*

1 Problem Statement

In this assignment you will implement a simplified version of *Distance Vector Protocol*. The protocol will be run on top of servers (behaving as routers) using UDP. Each server runs on a machine at a pre-defined port number. The servers should be able to output their forwarding tables along with the cost and should be robust to link changes. (Note: we would like you to implement the basic algorithm: count to infinity **not** poison reverse. In addition, the server should send out routing packets **only** in the following two conditions: **a) periodic updation and b) the user uses command asking for one**. This is a little different from the original algorithm which immediately sends out update routing information when routing table changes)

2 Protocol Specification

The various components of the protocol are explained step by step. Please strictly adhere to the specifications.

2.1 Topology Establishment

Each server is supplied with a topology file at startup that it uses to build its initial routing table. The topology file is local and contains the link cost to the neighbors. For all other servers in the network, the initial cost would be infinity. The topology file looks as follows:

- <num-servers>
- <num-neighbors>
- <server-ID> <server-IP> <server-port>
- <server-ID1> <server-ID2> <cost>

num-servers: total number of servers
server-ID, server-ID1, server-ID2: An unique identifier for a server
cost: cost of a given link between a pair of servers. Assume that cost is an integer value.
Consider the topology in fig 1. We give a topology file for the server 1

Line number	Line entry	Comments
1	5	number of servers
2	3	number of edges or neighbors
3	1 128.8.6.1 4091	server-id 1 and corresponding IP, port pair
3	2 128.8.6.2 4094	server-id 2 and corresponding IP, port pair
4	3 128.8.6.3 4096	server-id 3 and corresponding IP, port pair
5	4 128.8.6.4 7091	server-id 4 and corresponding IP, port pair
6	5 128.8.3.5 7864	server-id 5 and corresponding IP, port pair
7	1 2 7	server-id and neighbor id and cost
8	1 3 4	server-id and neighbor id and cost
9	1 4 5	server-id and neighbor and cost

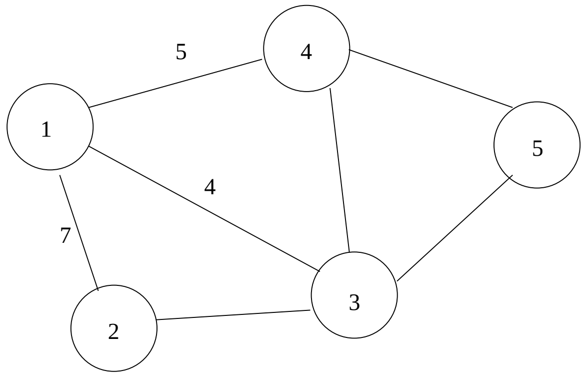

Figure 1: *Example topology*

IMPORTANT: Please adhere to this format for the topology file. In this environment, costs are **bi-directional** i.e. the cost of a link from A-B is the same for B-A. Whenever a new sever is added to the network, it will read its topology file to determine who its neighbors are. Routing updates are exchanged periodically between neighboring servers. When this newly added server sends routing messages to its neighbors, they will add an entry in their routing tables corresponding to it. Servers can also be removed from a network. When a server has been removed from a network, it will no longer send distance vector updates to its neighbors. When a server no longer receives distance vector updates from its neighbor for three consecutive update intervals, it assumes that the neighbor no longer exists in the network and makes the appropriate changes to its routing table (link cost to this neighbor will now be set to infinity but **not remove** it from the table). This information is propagated to other servers in the network with the exchange of routing updates. Please note that although a server might be specified as a neighbor with a valid link cost in the topology file, the absence of three consecutive routing updates from this server will imply that it is no longer present in the network.

2.2 Routing Updates

Routing updates are exchanged periodically between neighboring servers based on a time interval specified at the startup. In addition to exchanging distance vector updates, servers must also be able to respond to user-specified events. There are 4 possible events in this system. They can be grouped into three classes: **topology changes, queries and exchange** commands. Topology changes refer to an updating of link status(**update**). Queries include the ability to ask a server for its current routing table (**display**), and to ask a server for the number of distance vectors it has received (**packets**). In the case of the packets command, the value is reset to zero by a server after it satisfies the query. Exchange commands can cause a server to send distance vectors to its neighbors immediately. Examples of these commands include:

- update 1 2 inf
  The link between the servers with IDs 1 and 2 is assigned to infinity.

- update 1 2 8
  Change the cost of the link to 8.

- step
  Send routing update to neighbors right away. Note that except this, routing update **only** happen periodically.

- packets Display the number of distance vector packets this server has received since the last instance when this information was requested.

- disable server-id
  Disable the link to given server. Here you need to check if the given server is its neighbor.

- crash
  Emulates a server crash. Close all connections on all links. The neighboring servers must handle this close correctly and set the link cost to infinity.

- display
  Display the current routing table. The display should be formatted as a sequence of lines, with each line indicating: <destination-server-ID> <next-hop-server-ID> <cost-of-path>

2.3 Message Format

Routing updates are sent using the General Message format. All routing updates are unreliable messages.The message format for the data part is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Number of update fields   |            Server port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Server IP                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Server IP for 1                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Server port  1              |             0x0               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Server id for  1            |             Cost for 1        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Server IP Address 2                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Server port 2               |             0x0               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Server id 2                 |             Cost 2            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            ....                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- **Number of Update Fields:** (2 bytes) Indicate the number of entries that follow.
- **Server IP:** (4 bytes) IP of the server sending this packet.
- **Server port:** (2 bytes) port of the server sending this packet
- **IP Address n:** (4 bytes) IP of the n-th server in its routing table.
- **Server port n:** (2 bytes) port of the n-th server in its routing table.
- **Server id n:** (2 bytes) Server id of the n-th server on the network.
- **Cost 1:** Cost of the path from the server sending the update to the n-th server whose ID is given in the packet. There would be a sequence of the last three fields.

Note that First, the servers listed in the packet can be any order i.e 5, 3, 2, 1, 4. Second, the packet needs to include an entry to reach itself with cost 0 i.e. server 1 needs to have an entry of cost 0 to reach server 1.

3 Server Commands/Input Format

The server must support the following options at startup:

- server -p <port> -t <topology-file-name> -i <routing-update-interval> -d <server-id>

  port: Port number at which the server will run.

  topology-file-name: the topology file contains the initial topology configuration for the server

  routing-update-interval: specifies the time interval between routing updates in seconds

  server-id: id of this server

The following commands can be specified at any point during the run of the server:

- update <Server-ID1> <Server-ID2> <Link Cost>

  Server-ID1, Server-ID2: The link for which the cost is being updated.

  Link Cost: specifies the new link cost between the source and the destination server.

  Note that this command will be issued to **both** Server-ID1 and Server-ID2 and involve them to update the cost and no other server.

- step

  Send routing update to neighbors (triggered/force update)

- packets

  Display the number of distance vector packets this server has received since the last invocation of this information.

- display

  Display the current routing table. And the table should be displayed in a **sorted** order from small ID to big.

- disable <server id>

  Disable the link to given server. To do this close all connection to the given server

- crash

  Server exits, closing all connections. This is to simulate server crashes.

4 Server Responses/Output Format

The following are a list of possible responses a user can receive from a server:

- On successful execution of an update, step, packets, display or disable command, the server must display the following message:

  40 <SP> <command-string > SUCCESS <CRLF>

  where command string is the command executed. Additional output as desired (e.g., for display, packets, etc. commands) is specified in the previous section.

- Upon encountering an error during execution of one of these commands, the server must display the following response:

  40 <SP> <command-string> <Error message> <CRLF>
  where error message is a brief description of the error encountered.

<SP> denotes a space and <CRLF> is carriage return followed by line feed.

5 What to submit ?

You need to submit a working implementation of the project by the due date. The project would be graded based on the accuracy of your implementation. In order to test your implementation the TAs would setup a demonstration session for your project. A sign-up sheet to determine the demo schedule would be sent out later.