CS 640 Introduction to Computer Networks **Spring 2005**
http://www.cs.wisc.edu/~suman/courses/640/s05

Programming Assignment 2: Routing Protocol

Assigned: February 21, 2005 *Due: March 14, 2005, 11:59pm*

1 Introduction

Routing is the main process used by Internet hosts to deliver packets. Internet uses a hop-by-hop routing model, which means that each host or router that handles a packet examines the Destination Address in the IP header, computes the next hop that will bring the packet one step closer to its destination, and delivers the packet to the next hop, where the process is repeated. To make this work, two things are needed. First, routing tables match destination addresses with next hops. Second, routing protocols determine the contents of these tables

1.1 Routing Protocol

This is a software running on top of routers to determine the appropriate path onto which data should be forwarded. The routing protocol also specifies how routers report changes and share information with the other routers in the network that they can reach. A routing protocol allows the network to dynamically adjust to changing conditions, otherwise all routing decisions have to be predetermined and remain static.

1.2 Routing Tables

Internet hosts use routing tables to compute the next hop for a packet. Routing tables can take many forms, but here is a simple model that can explain most Internet routing. Each entry in a routing table has at least two fields - IP Address Prefix and Next Hop. The Next Hop is the IP address of another host or router that is directly reachable via an Ethernet, serial link, or some other physical connection. The IP Address Prefix specifies a set of destinations for which the routing entry is valid for. In order to be in this set, the beginning of the destination IP address must match the IP Address Prefix, which can have from 0 to 32 significant bits. For example, a IP Address Prefix of 128.8.0.0/16 would match any IP Destination Address of the form 128.8.X.X.

1.3 Routing Metrics

Routing metrics are a scoring system for routes a routing device knows. Metrics are calculated by routers to prioritize routes from best to worst. Routers use the metrics to select the best possible routes to a given destination. Metrics can include hop count (how many stops there are between here and the destination), media type (Serial, Ethernet, SONET etc.), availability (whether the machine is up or down) and several other factors including some set by the Network Administrator. A lower metric generally indicates a better route.

1.4 Dynamic Vs Static routing

- **Static Routing:** Static routing is not really a protocol, simply the manual entry of routes into the routing table via a configuration file that is loaded when the device starts up. Static routing is the simplest form of routing, but it is manual and does not work well when it has to be entered on a large number of devices. It is also does not handle outages or down connections well, as the manual entries will have to be changed manually to recover from such a loss of connectivity.

- **Dynamic:** Dynamic routing protocols are software applications that dynamically discover destinations and how to get to them. A router will 'learn' routes to all directly connected networks first. It will then learn routes from other routers that run the same routing protocol. The router will then sort through it's list of routes and select one or more 'best' routes for each destination it knows or has learned. Dynamic protocols will then distribute this 'best route' information to other routers running the same protocol, thereby extending the information on what networks exist and can be reached. This gives dynamic routing protocols to adapt to network topology changes or outages 'on the fly'. Once all routers know all routes to all destinations, the routing protocol has said to have 'converged'. If a user were to attempt to reach a destination before the routing protocol converged, the packet would eventually reach a dead end: a router which does not know how to get to the destination, and the packet would be dropped.

1.5 Distance Vector Protocol Vs Link State Protocol

- **Distance Vector Protocol:**This type of routing protocol requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. Hello and RIP are common D-V routing protocols.
- **Link State Protocol:**This type of routing protocol requires each router to maintain at least a partial map of the network. When a network link changes state (up to down, or vice versa), a notification, called a link state advertisement (LSA) is flooded throughout the network. All the routers note the change, and recompute their routes accordingly. This method is more reliable, easier to debug and less bandwidth-intensive than Distance-Vector. It is also more complex and more compute- and memory-intensive. OSPF and OSI's IS-IS are link state routing protocols.

1.6 Problem Statement

For this part of the assignment, you will implement a simple routing protocol on lines of *Distance Vector Protocol* to be run on top of servers (behaving as routers). The servers communicate with each other based on routes established via routing protocol updates. The server will run on a machine at a defined port number. The servers will communicate with each other using routes established by the routing protocol.

2 Protocol Specification

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

2.1 Topology Establishment

The server is supplied with a "topology file" at startup that it uses to build its initial routing table. This routing table table contains an entry for each server in the network. The cost of the links to neighbors can be determined from the topology file. For all other servers in the network, the initial cost will be set to infinity. The topology file looks as follows:

- ⟨ `num-servers` ⟩
- ⟨`server-ID`⟩ ⟨`server-IP`⟩ ⟨`server-port`⟩
- ⟨ `num-edges` ⟩
- ⟨`server-ID1`⟩ ⟨`server-ID2`⟩ ⟨`cost`⟩

  `num-servers`: number of servers

  `server-ID, server-ID1, server-ID2`: An unique identifier for a server

  `server-IP`: IP address of a server

`server-port`: port of a server

`cost`: cost of a given link between a pair of servers. Assume that cost is an integer value.

Following is an example of a topology file:

Line number	Line entry	Comment
1	3	number of servers
2	1 128.8.6.1 4094	server-id 1 and corresponding IP, port pair
3	2 128.8.6.2 4096	server-id 2 and corresponding IP, port pair
4	3 128.8.6.2 7091	server-id 3 and corresponding IP, port pair
5	3	number of edges on the topology
6	1 2 7	cost of edge 1 to 2 is 7
7	1 3 4	cost of edge 1 to 3 is 4
8	2 3 5	cost of edge 2 to 3 is 5

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 it's 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). 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 it's 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 (triggered update). 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 (triggered/force update) right away. Note that routing updates otherwise just happen periodically.

- `packets`

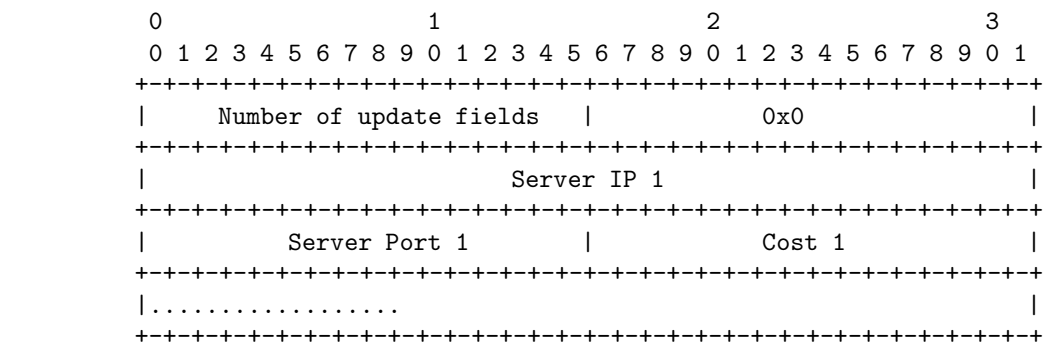
  Display the number of distance vector packets this server has received since the last instance when this information was requested.

- `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:



- **Number of Update Fields:** (2 bytes) Indicate the number of entries that follow.
- **Server IP 1:** (4 bytes) IP address of a server in the routing table of the server sending the update.
- **Server port 1:** (2 bytes) Port on which the server is running.
- **Cost 1:** (2 bytes) Cost of the path from the server sending the update to the server whose IP is given in the packet.

  There would be a sequence of the last three fields.

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`⟩

  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

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 function will be invoked at both `Server-ID1` and `Server-ID2` 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.

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 or display 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.

Submission

Questions and comments regarding this assignment should be posted to the mailing list, cs-640@lists.students.wisc.edu.