CSCI 651 Project 1

Introduction

In this project, you will implement a distance-vector routing protocol called RIPv2. Each node on the network executes RIP to exchange routing information with its neighbors, and based on this information, a node computes the shortest paths from itself to all the other nodes and the external internet. You will implement this routing protocol in Java. We will assume in this part of the project that the nodes cannot move, but may “disappear” and “reappear”.

The RIP Routing Protocol

You should first read the RIP specifications (RFC 1058 for RIP version 1 and RFC 2453 for RIP 2) VERY carefully, and understand them completely. The related RFCs can be found here: http://www.ietf.org/rfc.html . Be aware that you do not have to implement all the features defined in RFC 2453. Most of what we need can be found in Chapters 3 and 4. If there are any subtle issues about the specification, contact or e-mail the instructor.

Simulation

Since commandeering a lot of computers is impractical, we will have to make some concessions in the protocols and layouts in order to run all the software for all the nodes on one or more computers. The network will be simulated by assigning an IP and port to each side of a connection. The communication ports will be determined ahead of time (and in a high number range) for ease of use and avoidance of collisions.

Implementation

Your implementation at least should support 1) active RIP as a router, 2) handling incoming route messages, 3) CIDR, and 4) route message broadcasts. We assume that only RIPv2 is used. Also, set the route update time to 5 seconds, not 30 seconds as defined in the RFC, to reduce the convergence time, and 10 seconds for a node to be determined as offline. If a node fails, other nodes should respond to recover routes if alternate routes are available. To see this effect clearly, you can implement node failure by terminating the node's process. The neighbors of a failed node should detect that the failed node is indeed unreachable and set the appropriate distance to infinity. This information will be propagated to the entire network through the RIP routing protocol.

As you probably know, triggered updates need to be invoked for fast recovery, and split horizon with poisoned reverse needs to run to prevent the count-to-infinity problem.

Running Your Program

Your program should take as a command-line parameter the configuration file. To make startup (and life) a little easier, each node should broadcast on schedule to all neighboring networks. An example config file:

```
LINK: 127.0.0.1:63001 127.0.0.1:63002
LINK: 127.0.0.1:63003 127.0.0.1:63004
LINK: 127.0.0.1:63005 127.0.0.1:63006
LINK: 127.0.0.1:63007 127.0.0.1:63008
NETWORK: 10.0.1.0/24
```

Where "LINK" can be thought of as a cable between two nodes. The first address is the current node's address on the cable's subnet, and the second number is the node's neighbor. "NETWORK" is a network that is attached to this node (there may be multiple networks attached to a node). Each time the routing table changes internally, the node should print its routing table. An example output may be:

```
Address                 Next Hop                Cost
================================================
10.0.1.0/24             0.0.0.0:0               0
10.0.2.0/24             127.0.0.1:63002         1
10.0.6.0/24             127.0.0.1:63002         3
10.0.3.0/24             127.0.0.1:63004         1
10.0.4.0/24             127.0.0.1:63006         1
0.0.0.0/32              127.0.0.1:63008         5
```

Note 1: Example only, your results may be displayed in a different order.

Submission

Due date: Friday, October 13, 2017

Zip all source files, project files, makefiles, and anything else required to build a node jar file on a CS Ubuntu lab machine. Include a README.TXT file giving your name, RIT ID, and any build/run instructions. Submit the zip file to the MyCourses Project1 folder for this class.

Note: The network traffic between the nodes will be monitored and compared to the actual RIPv2 specification.