Programming Assignment: CE 156

This programming assignment focuses on socket communication between processes to achieve network routing.

Description:

In this assignment you will simulate a network. You will start independent and separate processes to represent each node in a network. Each node will read a file, *neighbor.config,* to determine its neighbors. The nodes then periodically exchange routing information with their neighbors and run Bellman Ford algorithm to compute a routing table.

The routing table should be printed with a timestamp whenever there is a change in the table. Test your program with a variety of network configurations and double-check your computed routing table against what you manually calculate. Bring nodes down by killing the process for the nodes and check that the routing information propagates correctly. Likewise, make sure that when you bring a node back up that the routing information propagates correctly.

RFC 1058 (http://tools.ietf.org/html/rfc1058) gives the specification for the Routing Information Protocol version 1.

Configuration file format:

There are two configuration files: node.config and neighbor.config. The format of the files is the following (the entries are just for illustrative purposes):

A node will use node.config to construct a bidirectional translation between virtual and physical addresses. This translation supports the simulation of the "virtual" network.

node.config:

```
# fields separated by spaces only (no tabs)
#<virtual IP>   <virtual port>   <physical IP address> <physical port>
10.0.0.1          520             192.168.0.1    4021
10.0.0.3          520             192.168.0.3    4023
10.0.0.5          520             192.168.0.5    4025
10.0.0.7          520             192.168.0.7    4027
```

A node uses neighbor.config to know which other nodes are directly connected to it.

neighbor.config:

```
# fields are separated by spaces only (no tabs)
# <virtual IP>     <virtual IP>   <distance>
10.0.0.1           10.0.0.3          30
10.0.0.1           10.0.0.5          50
10.0.0.1           10.0.0.7          70
```

Sample set of commands:

Open four terminal windows. Each line is executed in a separate terminal window.

terminal1> rserver 10.0.0.1 520

terminal2> rserver 10.0.0.3 520

terminal3> rserver 10.0.0.5 520

terminal4> rserver 10.0.0.7 520

Be prepared for nodes from configuration file to die and come up again randomly.

What to submit?

You must submit all the files in a single compressed tar file (with `tar.gz` extension). The files should include

1. A specification of the application-layer protocol, describing the handshakes involved, message formats, error handling, etc. (Note: you need to describe only the handshakes above the socket layer, not the handshakes within the TCP layer).
2. A Makefile that can be used to build the client and server binaries.
3. A README file including your name and a list of files in the submission with a brief description of each. If your code does not work completely, explain what works and what doesn't or has not been tested.
4. Documentation of your design in plain text or pdf. Do not include any Microsoft Word files. The documentation should describe how to use your application and the internal design of your client and server implementations.
5. Organize the files into directories (*src*, *bin*, *doc*, etc.)

Grading

Each submission will be tested to make sure it works properly and can deal with errors. Grades are allocated using the following guidelines:

Basic Functionality:	50%
Dealing with errors	20%
Documentation	20%
Style/Code structure, etc.	10%

Note that 20% of the grade will be based on how well your code deals with errors. Good practices include checking all system calls for errors and avoiding unsafe situations such as a buffer overflow.

The files must be submitted before midnight on the due date.

Honor Code

All the code must be developed independently. All the work submitted must be your own.

Deliverables: This part of the assignment is due on: ____Mar. 16_____

Simplified RIP Version 1 packet format: