Kevin Milczewski, Eitan Adler
CS428 Lab 3

Program use

The config file for our project is found in config.c and .h. The number of hosts must be updated at the top of config.h and then fill in the hosts in config.c. Example hosts are provided. Neighbors don't have to be symmetric, but should be for easy reading of the file. Note that the MAX_HOSTS macro must match the number of hosts provided, this should be fixed in the final build.

A test scripting file is included. It will launch the servers by number and record their output to files named “servout#.txt” where # is the number of the host. The PID of each process is also recorded into the file kil_list.txt. Note that the number of hosts launched by the test script must be updated independently of the config file, and care should be taken that you're executing the servers on the proper machines.

To stop the servers gracefully, our servers exit on SIGHUP. The shell command < pkill -1 server > should be used to stop the servers. The PID file can also be used to stop the processes.

If the system you're compiling our code on does not have the clang compiler, an alternative makefile has been provided that should compile the code on G7. It's a file called g7make, simply replace the Makefile with this. All it does is change the compiler to gcc and add a needed flag.

Program structure

Config.h/c: Contains configuration information about the server setup, mainly the host arrangement. The initconfig() function runs before main through the use of a constructor attribute.

Debug.h/c: Has some basic debugging macros and functions.

Util.h/c: Contains utility functions like die and getSocket.

Packets.h/c: Defines the structure of a packet header as well as the maximum data size and TTL of packets.

Routing.h/c: Defines the structure of the routing table, as well as the infinity representation for distance and the max ttl of a routing table entry. The function init_routing_table() initializes the routing table from the configuration file. This is the only time the host structure in config.c is accessed.

Server.c: This is the core of the program, and has the main function. The primary methods to look at are the pthread functions, routingthread and timerthread. These threads run concurrently, timerthread handles updating the ttl of routing table entries and sending the table to neighbors, while routingthread handles receiving tables from neighbors. Ignore forwardingthread for now, it is not yet implemented completely.

Routing Algorithm

We used a simplified version of path vector for our routing algorithm. Each router keeps a list of nodes that will be taken to get to the destination of the row in the table. The entire table is sent to neighbors when communicating, then if a node finds itself on the list of nodes sent to it, it does not update the table. The exception to this is when the table came from the next hop to the destination.

Our algorithm can automatically discount nodes when they go down, and add them back when they go back up. Adding a connection is a trust based action: If a server advertises a zero length distance to itself, the current router believes it and adds a direct link to it.

We have a short todo list for improving our implementation of the algorithm. When a node receives an update from another node, it can update all it's other neighbors of the change immediately, but our algorithm waits for the timer thread to fire. This would reduce the time to react to changes in the topology. The immediate update is on the todo list.