Part 1 - User Guide

Introduction

This implementation for Part 1, based on the Client-Server architecture described in the Part 1 project specs, is a basic message-passing system. A pair of nodes, communicating over a TCP link, uses a server-router, S-Router, to appropriately redirect communication.

Both the client nodes (TCPClient.java) and server nodes (TCPServer.java) make the initial connection to the server-router (TCPServerRouter.java) using the TCP 5555 port. After that they get assigned a random port and each of them gets a separate thread (SThread.java) so that the router can stay connected to multiple clients/servers. There is a 10 second wait after each connection is made (in each thread) in order to allow enough time for all nodes' information to be inserted into the routing table.

The communication procedure is rather simple. Each client program reads lines from a file (file.txt) and sends them individually to the server-router, which passes them to the server. Server converts each line it receives to upper case and sends it back to the client the same way. Each node shows what is going on in its output, but only the client calculates the summary data on each cycle (in milliseconds) of execution, presenting the results after each upper-cased string is returned. The last line in the text file (Bye.) terminates the simulation.

Configuration

In order to correctly run the simulation each client/server/router ought to be placed on a different machine. The server-router will require both files (TCPServerRouter.java and SThread.java) in the same directory and each client will need the text file (file.txt) in its directory as well (different text files may be used for distinct clients in order to have diversity in the simulation). The number of client/server pairs depends on the length of the routing table (in server-router file), which is its length divided by two; however, it must be kept in mind that multiple copies of clients/servers should not be opened from the same location (like an H drive on the network) in order to allow different content in the files, because the clients will be using distinct IP addresses for their destinations.

A program like jGRASP would be appropriate to use for compiling/executing the files; but first, all instances of clients and servers need to have appropriate setup. First, the machine name for the server-router should be entered. Line 13 (in both files) has the variable: `String routerName`, which is the name - a symbolic name labeled on the machine or assigned to it. It can be found by looking up the machine name on the computer from which the server-router will run (right-click My Computer icon and select Properties, then look in Computer Name tab). Do the same look-up if you are using your (wireless) laptop in the configuration.

Second, the IP addresses for the corresponding clients and servers should be entered. Line 36 in the client and line 34 in the server have the variable: `String address`, which is the IP for the destination to which the node will send messages. It can be found by typing “ipconfig” in command prompt on the destination computer (press Run in Start menu, type cmd, press OK, type ipconfig, and press Enter; the IP to use is to the right of: IP Address). Once these steps are complete, the simulation is ready.

Compilation

All the files should be successfully compiled before any execution. Assuming that different clients/servers are not opened from the same location, each pair will be different and will work independently from others.

Execution

The server-router should be executed first; it will be listening on port 5555 for incoming connections. The client/server pair should be executed as closely (time wise) as possible; enough time for them to be ready to accept messages from each other, after the router has them wait for 10 seconds. It doesn’t matter at what point each separate pair is executed, since none of them interact. Depending on the length on the text file(s), each of them may take varying amounts of time to complete.