Project 1 | Phase 1

Distriuted Computing

Dhruval Darji

Deion Anthony

Mirela Memic

Table of Contents

[Basics: 3](#_Toc433720740)

[Process: 4](#_Toc433720741)

[System Process Steps: 5](#_Toc433720742)

[System Modules: 6](#_Toc433720743)

[Output: 7](#_Toc433720744)

[Data: 8](#_Toc433720745)

[Conclusion: 9](#_Toc433720746)

# Basics:

For Phase I of this project we implemented a file text transfer service using TCP protocol. The purpose of this project is to transfer data from a client to a server via a server-router and then receive a response from the server to the client. We used four main modules to implement this scenario. The modules are named TCPServerRouter, SThread, TCPServer, and TCPClient. Additionally, we used a custom statistics module to monitor and keep track of variable data. Variables include amount of data sent versus the amount of data received, as well as the time it takes to send data and receive a response. This module was called Stats, and it was built to be invoked on the TCPClient and TCPServer modules. Finally, we used a main module to dynamically run either the TCPServerRouter, TCPServer, or TCPClient modules from any machine without having to drastically change code for variable inputs such as IP addresses. We also implemented a feature in the main module to enable running the client and server modules continuously for data collection purposes. This module was called Main. More detail on each module can be found in the System Modules section.

# Process:

# System Process Steps:

In order to run our processes, we need at a minimum 2 machines, though using 3 is preferable. This is because the TCPServerRouter module can be run on either machines that are planning on running the TCPServer or TCPClient modules. In this explanation we will be using 3 machines. To start our process, the Main module on each machine should be updated with the IP address and port number of each other machines and their respective roles. Then machine 1 should run the Main module and input ‘r’ to run as the TCPServerRouter. Next, machine 1 and 2 should run when they see fit their Main modules. They should then input ‘s’ for the TCPServer module and ‘c’ for the TCPClient module. Note that the machine running the TCPClient module must always specify the location of the text file they wish to read from and send to the machine running the TCPServer module. Now we get into the details of each module.

When the TCPServerRouter module is run it first creates a RoutingTable to store clients/servers that connect to it. This RoutingTable is a HashMap with String Keys and Socket Values. This enables multiple server/client connections to the TCPServerRouter that are mapped to their respective sockets. This makes searches to those addresses simple and retrievable in constant time. Additionally, addresses that have previously connected won’t be marked as new connections, but will instead simply have their respective socket replace the old unused socket. Next the module creates a ServerSocket for accepting connections on a pre-specified port number. By default, we use port number 5556. Finally, for every instance of a client/server connecting it creates an instance of the SThread module to handle the connection between a Server and a Client. It then passes in an instance of the RoutingTable and the connecting client to that instance of the SThread module. The TCPServerRouter module only disconnects and shuts down if it’s Running variable is switched to false, or if it is forcefully terminated. This is how the TCPServerRouter module runs.

When the SThread module is invoked by the TCPServerRouter module it instantiates new PrintWriter’s, InputStreams, and OutputStreams using the socket that was passed to it it from the TCPServerRouter module. Next it takes the RoutingTable instance that was passed to it from the TCPServerRouter module and updates the RoutingTable, by setting the Key to be the IP of the server/client and the Value be the new socket that was also passed to it. The next step occurs once the TCPServerRouter module invokes the Run method of the SThread module. When this is done the SThread module starts listening for incoming messages that contain the destination the client/server wish to connect to and prints out to notify the client/server that a connection to the router has been made. Then it sleeps for 10 seconds to allow the RoutingTable to update and for changes to take place. Then the module searched the RoutingTable to find an instance of the destination that the client/server is seeking. Once the destination is found, an instance of a new outSocket or Output Stream is created to communicate with that destination. Then, while there is still input the SThread module will do two things. The first thing is to check if the input was a termination sequence. If it was, then it will print the termination sequence to the destination and then terminate. Otherwise, if the destination socket still exists, it will print the incoming message to the destination. This is process of the SThread module.

The next module we would like to discuss is the Stats module.

The TCPServer …

The TCPClient …

# System Modules:

# Output:

# Data:

# Conclusion: