435 Course Project Report

Option I (Client-Server)

## a. Design Challenges and Model Assumptions Made

I had initially thought about creating multiple client GUIs to interface with the server. In such a scenario, I would have to rapidly switch between Client GUIs to enter data to the server. I did not think this would reflect the real-time nature of a client-server network and it would also make testing the system highly arduous for the TA.

From this I decided to automate client requests using a sleep function to simulate realistic pauses of 1 to 4 seconds between requests from clients. In addition to automatic requests, I had them generated randomly. By doing this, I incurred the added benefit of being shown several bugs that I would not have otherwise found.

As a result, the final system is more of a simulation of a client-server system than an intractable one, but allows for the user to clearly observe the logical inner workings of the client-server network.

## b. Difficulties and Resolutions

Major difficulties

1. Establishing client attributes and separating client-server functionality
2. Storing and Accessing information about peers

Resolutions

1. Initial thoughts were that the client needed to store all of its information (name, status, number of requests). It became clear however that such information was only used by the server. Therefore I created the ServerWorker class, separate from the server class, to handle the operations specific to handling client requests.
2. It was also unclear where and how to store peer information and server history, which is how the RequestData class was created (to store request information on the server).

## c. System Model

The system starts in the Main class, which runs the main(String[] args) method of the program, starting the server and the clients. Once created the server simply waits for connections from clients. Once connected to the server, a ServerWorker is created to handle all of the client’s requests. Any requests are stored in the RequestData class and stored on the server to be retrieved by platinum clients who want to monitor the network’s data.

For every client that connects to the server, a ServerWorker object is created to take care of all that client’s needs, as well as record its requests to the server.

The system is a client-server application that was implemented with TCP connections in mind. This is because it was necessary for all messages between client and server be registered. It would be unacceptable, for example, for one of a client’s allotted request tokens to be used up by the server, only for the requested web page to be lost.

The system runs until a certain number of requests have been submitted by all clients or when all are disconnected from the server.

## d. Future Work

Improvements to the current program include:

* Implementing additional thread control using semaphores. Specifically to pause output of other clients while trying to output network usage details for platinum clients. In retrospect, having 3 separate client GUIs would have reduced the significance of this problem by having servers.
* Currently, the system only checks for clients number of requests before carrying out a request. Ideally, should check again afterwards and disconnect client immediately if remaining requests is 0, to connect waiting clients to the server sooner.
* Formatting of Network Usage Details could be improved using java’s formatting library instead of manually inputting spaces into strings.

## e. Compile and Run

1. In terminal/command line, change the directory to the one containing the .java files.
2. Enter “javac Main.java” to compile the program
3. Expand the width of your console to improve formatting clarity
4. Enter “java Main” to execute the program
5. Enter “ctrl+c to” end the program.