Program Operation

TCP

To make a TCP connection, the server has to be running first. When the server starts up, it claims a port number on the machine based on user input and becomes available via it’s machine’s IP address. The server does so by creating a ServerSocket with the given parameters. Then, the ServerSocket waits for connection requests from clients. When a client starts up, it receives user input about where to attempt a connection. The client then makes a Socket object and points it toward the address entered by the user. If there is a ServerSocket waiting for connections at the given address, then it can accept the connection request from the client. This creates an open connection between the client and the server. When the client sends a message over the connection, the TCPServer class receives the input over the input stream from the ServerSocket. In this case, when data is received over the input stream, the Server echos the input over the output stream of the ServerSocket. In the case of TCP, the server can connect to multiple clients at the same time. To achieve this, the Server class creates a new Thread for each connection that is made, and each Thread handles information from one ServerSocket connection.

UDP

In order to utilize the UDP protocol between a client and server, the server must be provided a port number to use the DatagramSocket class. DatagramSocket creates a socket that allows for the server to listen for DatagramPacket(s) that are sent via the UDP protocol on the given port. Likewise, for the client, it starts a DatagramSocket class that also creates a socket. Unlike the UDP server, the client socket does not require a port number, however, it needs a hostname or IP address to know where to send the DatagramPacket(s) to. There is no ‘connection’ between the UDP client and server because the UDP protocol is intended to overlook the establishment of a connection. However, in order for the client to send data to the server, the client must know the hostname or IP address of the running UDP server. The request is then received by the UDP server after which the server will then send back a DatagramPacket response.

Differences

The main differences between TCP and UDP in this implementation are the connections and the use of Threads associated with them. TCP requires connections to function, while UDP does not. Since TCP needs to create connections, it requires the use of Threads to manage multiple connections at a time. Since UDP does not require connections, the server can handle multiple connections without needing to create multiple Threads. TCP also deals with a three-way handshake, while UDP does not. We did not implement this in our project because it is built into the Java libraries we used.