Network Programming Assignment 1

Tong Chen

10189689

This assignment uses 3 classes: XServer, Client, and WorkerThread.
XServer creates a server socket for any clients to connect to. It holds a counter for how many clients are currently being handled. It also creates a thread pool to limit the number of threads running to 3. The server will continuously listen for connection requests made by clients. When a connection to the server is requested, the client is assigned a name and an input and output stream is created to the client socket. If there are already 3 clients connected to the server, the server will close the connection. Else, the server will create a new WorkerThread using the client socket, name, input stream, and output stream.

Client requests a connection to the server and creates an input and output stream to the server socket. It waits for a reply from the server to determine if the server can handle the client. If it cannot, the client will close its connection with the server. Else, the client can begin to send messages to the server. If “list” is sent, the client will continuously listen for messages from the server until the server lets the client know that all the files have been transmitted. If “exit” is sent, the client and server will both close the connection to each other. Else, the client will print the acknowledgement message from the server.

The WorkerThread class handles a single connection with a client. It overrides run() in order to run out own logic when a new thread is created. When the new thread begins execution, it will increase the client counter by one, record the time at which the connection was accepted, and sends the client a message letting it know that its connection was accepted. The WorkerThread will then listen for messages to be sent by the client. If “exit” is received from the client, the thread will close the connection to the client, decrease the client counter by 1, and end execution of the thread. If “list” is received, the server will list all the file names in the repository and send a final message letting the client know that all the files names have been listed. Else, the server will return the message with ACK appended to the end of it.

One of the difficulties faced was making the server multithreaded. It stemmed from using thread pools because the thread pool would queue up any threads beyond the set limit. This would cause the undesired effect of leaving clients hanging while it waited for to be placed in the thread pool. This was handled by using a client counter and placing an additional check before creating a new WorkerThread. The result was closing the connection to the client and send the client a message letting it know that the connection could not be accepted so the client does not have to wait.

Some improvements I would make would be to add the streaming of files from the server to the client. I would also let the server admin enter the path to the repository rather than hard coding it.

Another method besides using thread pools would also be possible to avoid the use of a client counter to make the project more thread safe.