Sam Buahin
Myschan Vajh
ITCS 6166-001

Project 1 Report

Objectives

The goal of Project 1 is to develop working HTTP client to server functionality through a TCP connection that has two commands: get and put. Get will allow the client to send a request to the server and the server will respond by returning the requested object where the client will then save it locally in the current directory. Put has the client request that the server store a file and the server will respond by saving it locally. Detailed below are the methods taken to achieve this goal.

Summary

When the Server is started, it requires the user to enter a port number. The clients requests for a host, port number command and filename.

The programs send all this initial input to the Server which first checks if file exist. Both Client and Server uses control statement to decision on which method to call.

Design Overview

The programs written to complete Project 1 were written in Java via the IntelliJ IDE. There are two primary files, HttpClient and HttpServerNew. HttpClient contains four methods: main, sendFile, and receiveFile and fileType(). HttpServerNew has six methods: main, HttpRequest, run, receiveFile, fileType() and sendFile.

HttpClient: main() method

Sam Buahin
Myschan Vajh
ITCS 6166-001

HttpClient sends requests to the server regarding file transfers. The main method begins by requesting that the user enters 4 parameters. These parameters are the hostname, the port number, the command, and the name of the file. The socket myclient is created using the hostname and port while the command and file names are sent through it to the server by a data output stream. It then waits for a response from the server via a date input stream to confirm the requests were received. Main then runs its get or put method based on the users input and closes the socket reader and writer afterwards. This is all encapsulated in a try-catch block with appropriate error handling.

HttpServerNew:

main method HttpServer begins by asking the user to enter a port number to be used. This port number will allow it to be found by clients who use the same number. A server socket is created and it waits for a connection to be made before continuing. Once a connection is accepted, the program initiates the HttpRequest method.

HttpServerNew: HttpRequest and run methods

HttpRequest acts as the Thread in the program. Whenever a new client connects to the program, a new Thread n created. When the clients finished what is doing and closes, the Serve continues to run in the while(true) loop unless CTRL-C key combination is held.
The command and file name are read from an input stream. The properties of the file are found through the client directory. Once this is all successfully completed, it writes a message to the

client that the client prints out through a data output stream, confirming that the request has been received. It then gets or puts based on the command and announces that the client connection has closed afterwards while staying open to connect with other clients.

sendFile Algorithm

sendFile works by creating a byte array with dimensions matching the file size. A file input stream is processed through a buffered input stream to send it to the server with minimal data loss or corruption. The buffered input stream reads in the byte array and sends it through the socket's output stream to the server. Once this is all flushed, the streams and sockets are closed.

receiveFile Algorithm

recieveFile also creates a byte array with the same dimensions while creating input, output, and buffered output streams. The input stream reads from the server and is used to calculate the size of the transfer. The output stream is once again processed through the buffered stream. An integer variable current is used to iterate through the byte array as the information is copied over in a do-while loop. The buffered stream flushes this and all the streams and the socket are closed afterwards.