CSS432 A

Documentation – Homework 2

Simplified HTTP Retriever and Server

Johnathan Hewit

# **Design/Algorithm**

## Server

The server application is designed to run a main function that opens a connection socket and listens on a port defined in the top of the source file (presently coded to be the last four of my student ID) on execution. Upon listening, it opens a new socket descriptor for its incoming client, and loops infinitely to accept incoming connections. Once an incoming connection is made, it threads it and calls a function to reads its contents and parse the HTTP request it’s expecting. To accomplish this, character pointers are created and assigned to each of the relevant pieces of the HTTP GET request.

Once it reads the path of the request, it determines if the file being requested exists locally, and sends the file to the client by creating a buffer and streaming the file into by chunks into the buffer. If the client requests a file that is not found, requests a restricted file, or attempts to access other directories, the server responds to the client with the corresponding HTTP error code. Once the response is sent, the server closes the socket with the client, and continues to listen on the specified port for additional requests from clients.

## Retriever

The retriever application is designed to run a main function that receives 3 initial arguments on execution of the executable. Each argument is entered in the following order: hostname, path to file, and port number for the server. The retriever does some validation checking to ensure it received correct information required to resolve the hostname to an IP and attach to a GET request. Namely, it looks for [http://](NULL) or [https://](NULL) and strips them off the hostname if they’re found. Next, it creates the socket and assigns it to the retriever’s socket descriptor (SD). Next, it attempts to make a connection with the server’s information it was provided and attaches the SD to the socket. Once the connection is made, it forms the HTTP GET request with the information provided and sends it to the server and awaits the response.

Once the server responds, the retriever then reads the response by calling a function to parse the header. The function looks for the “200” code response from the server indicating the request was processed for the requested file by scanning the first line of the header. If the retriever does not find the 200 code, it prints the HTTP header to the console. If it does find the 200 code, it prints the HTML’s contents to the console and calls a function to save the file to the root directory of the project. Once the function completes, the retriever closes the socket and the program ends.