Design Document: Mulithreaded WebDownloader and Http Server

Aaron Nguyen
CruzID: anguy200

1 Goals

The goal of this was to create a web downloader that is multithreaded to connect to multiple servers to download chunks of a file at a time. In addition a server was also required to meet these needs. The server did not need to be multithread, but the server I made is.

2 Design

To achieve this code from Lab 1 was used as an foundation and more was added to differentiate the client and the server. In addition a C library called pthreads was used to make the client and server multithreaded. In order for the client and server to communicate a simplified version of the Http Request and Response was used in the connection for requests and responses. Errors handling is primarily handled by the server in the form of error codes like 400, 403, 404 and 500, in case of issues with connection and server. The error handling for the client was roughly similar, but primarily focused towards handle errors internally rather than externally like the server. Just in case the Client also has http error codes like the server. A specific part of http used was the inclusion of "Content-Range: #-#/#" in the request and response for GET. This allowed for files to be downloaded in chunks.

Functions for Server and Client

2.1 Main Function

In this function it parses through the argument line finds the hostname, ip, port, and filepath. It also creates the HTTP requests and establishes a connection with a requested server. It also receives and records information sent from the server to either stdout or a file called output.dat. In addition, the program uses a helper function called catch file length to determine when to end a connection with a server. In addition it also initialized threads and depending on client or server it would also open up a list of servers and start parsing the file. In addition a Producer and Consumer methodology was used to multithread the client and server.

2.2 Catch Length/Catch Range

This purpose of this function is to take the header function recorded by the main function and parse through the string to find the Content-Length of the file that is going to be sent to the program. If the function finds a content length it will return the length number. If not the function will return a -1.

2.3 Error_Print

Contains all HTTP error responses and codes and send out to connection if conditional is reached.

2.4 Client - HTTP Requests

A function that contains strings and when called sends specific strings to server to make a GET or HEAD request.

2.5 Client - Head_Parse/ Get_Head

Parses and requests header information from server. Primary purpose of this function is to get the length of the requested file.

2.6 Establish_Connection/ Parse_Recv

Are the actual main functions of the program. Each thread uses these functions at start, and from there makes requests/responses. These functions call helper functions read and write to local files and sends data over the connection.