CSC 262 Operating Systems Course
Final Project Requirements

Multi Threaded Web Server with an Admin Console

Objectives

There are four objectives to this assignment:

- To modify an existing code base in Java
- To learn how to create and synchronize cooperating threads in Java.
- To gain exposure to how a basic web server is structured.
- To think more about how to test the functionality of a code base.

Starting Point

In this assignment, you will be developing a real, working **web server.** To greatly simplify this project, we are providing you with the code for a very basic web server. This basic web server operates with only a single thread; it will be your job to make the web server multi-threaded so that it is more efficient.

Background

HTTP (hyper text transfer protocol) is the main protocol between web servers and web browsers. HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the *client* and an application running on a computer hosting a website may be the *server*. The client submits an HTTP *request* message to the server. The server, which provides *resources* such as HTML files and other content, or performs other functions on behalf of the client, returns a *response* message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

A web browser is an example of a *user agent* (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them when possible to reduce network traffic. HTTP proxy servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP is an application layer protocol designed within the framework of the Internet protocol suite. Its definition presumes an underlying and reliable transport layer protocol,[2] and Transmission Control Protocol (TCP) is commonly used. However HTTP can be adapted to use unreliable protocols such as the User Datagram Protocol (UDP), for example in HTTPU and Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes *http* and *https*. URIs and hyperlinks in HTML documents form inter-linked hypertext documents.

HTTP/1.1 is a revision of the original HTTP (HTTP/1.0). In HTTP/1.0 a separate connection to the same server is made for every resource request. HTTP/1.1 can reuse a connection multiple times to download images, scripts, stylesheets, *etc* after the page has been delivered. HTTP/1.1 communications therefore experience less latency as the establishment of TCP connections presents considerable overhead.

Basic Web Server Codes

Codes are available at the web page of the course or you can simply download the codes from the following link : http://sadievrenseker.com/wp/wp-content/uploads/2016/08/server.java_.zip
In the class, we have also covered the details of the coding and tested web server with the browsers (like chrome or safari).

Status Codes

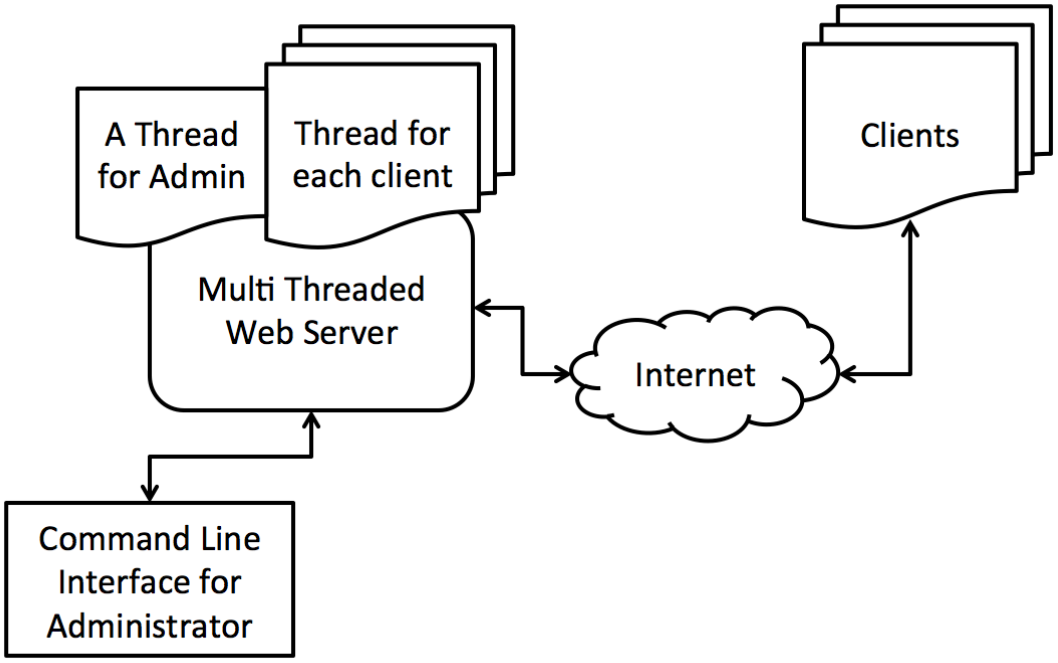
In the starting code parts, you will find the HTTP status codes. You can also check the full list of codes from Wikipedia (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

Requirements

The project can be divided into two major parts:

1. Administrator console interface
2. Web Server Functionalities

An overview of the system can be demonstrated as below figure



Your server will support multiple connections from the clients. Each client connection request will be handled with a separate thread on the server. Also there will be a command line interface (CLI) for the administrative commands and a separate thread will handle the request.
You can use web browsers, or you can implement your own client in Java.

Admin CLI Requirements:

1. Starting server : You can use a simple command like “start”, to start the server and accepting new connections from the clients
2. Shutting down: You can use another command like “shutdown” to stop the server. In order stop server, you must be careful about stopping all active threads.
3. Listing active connections (optional but would be required for debugging your code): just dump all the active server threads and some details about them for debugging.

Multi Threaded Server Requirements:

1. You must provide a thread for each active connections
2. Serve the requested file from the file system of the server.
3. Keep connection active until client requests a close
4. Use the communication through HTTP (already provided some source codes)

Optional Implementation (strongly suggested): Write your own client with own command sets. You can also use a browser, like chrome or safari but this will make coding on the server side a little bit more complex.

Submissions

For each of the project phases, please provide first two items below. For the final project submission you should also demonstrate the last item in the list.

- Return the source codes you have implemented for the project
- Write a brief report about your implementation details, your assumptions (if you did any)
- Demonstrate your project a running scenario, with more than 3 connection request at least at the same time and their views on the server. You need to find a way of demonstration (like the "list" command from the admin CLI)

Project Phases

Project Phase #1 : Writing multi threaded server: Modify the web server code provided in the lab session. The new version of the server will accept multiple connection requests at the same time.

Project Phase #2: Provide a level of synchronization. You have a multi threaded server from the first phase. Now consider both clients are trying to read the same file and you can allow only 1 access at a time. Provide a synchronization for such a case

Project Phase #3: Implement the administrator module with the requirements above.