Programming Project Example 4

1 Introduction

Berners-Lee and his team are credited for inventing the original Hyper Text Transfer Protocol (HTTP) along with Hyper Text Markup Language (HTML) and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page. What you're about to do is to reinvent the wheel on the motivation of getting a deep understanding of how HTTP works!

In this assignment, you will use sockets to implement a simple web client that communicates with a web server using a restricted subset of HTTP. The main objective of this assignment is to give you hands-on experience with "UNIX" sockets.

2 Multi-threaded Web Server

2.1 Introduction and Background

Please refer to the lectures' slides for the format and the use of HTTP.

2.2 Specifications

Your web server should accept incoming connection requests. It should then look for the GET request and pick out the name of the requested file. Note that a GET request from a real **WWW** client may have several lines of optional information following the GET. These optional lines, though, will be terminated by a blank line (i.e., a line containing zero or more spaces, terminated by a '\r\n' (carriage return then newline characters). Your server should first print out the received command as well as any optional lines following it (and preceding the empty line).

The server should then respond with the line, this is a very simple version of the real HTTP reply message:

```
HTTP/1.0 200 OK\r\n
then in case of GET command only:
{data, data, ...., data}
```

followed by a blank line (i.e., a string with only blanks, terminated by a '\r\n'). After finishing the transmission, the server should close the socket created by the accept() function and wait to accept a new connection request. If the document is not found (in case of GET), the server should respond with(as would a real http server) :

```
HTTP/1.0 404 Not Found\r\n
```

2.3 Server Side Pesudo Code

```
while true: do
    Listen for connections
    Accept new connection from incoming client and delegate it to worker thread/process
    Parse HTTP/1.0 request and determine the command (i.e. GET)
    Determine if target file exists (in case of GET) and return error otherwise
    Transmit contents of file (reads from the file and writes on the socket) (in case of GET)
    Close the connection
end while
```

2.4 Notes

- You should handle the both commands GET (to get file from the server).
- No validation (on requests) required. However, you should write the described format of the command.
- There are different types of HTTP status codes, you're only required to handle 404 and 200.

- You will want to become familiar with the Socket programming in Java(check out Oracle Socket Tutorial: `https://docs.oracle.com/javase/tutorial/networking/sockets/`.
- You are supposed to handle **HTML, TXT and images**.

3 Bonus

3.1 HTTP 1.1

If a web page contains 4 images, a total of five separate connections will be made to the web server to retrieve the html and the four image files. Note that the previous discussion assumes the HTTP/1.0 protocol which is what you will be supporting in this first assignment.

Next, add simple HTTP/1.1 support to your web server, consisting of persistent connections and pipelining of client requests to your web browser. You will also need to add some heuristic to your web server to determine when it will close a "persistent" connection. That is, after the results of a single request are returned (e.g., index.html), the server should by default leave the connection open for some period of time, allowing the client to reuse that connection to make subsequent requests. This timeout needs to be configured in the server and ideally should be dynamic based on the number of other active connections the server is currently supporting. That is, if the server is idle, it can afford to leave the connection open for a relatively long period of time. If the server is busy, it may not be able to afford to have an idle connection sitting around (consuming kernel/thread resources) for very long.

3.2 Test your server with a real web browser

Test your server with a web browser of your choice.

4 List of Useful Resources

- Oracle Socket Tutorial: `https://docs.oracle.com/javase/tutorial/networking/sockets/`
- KKMultiServerThread.java: `https://docs.oracle.com/javase/tutorial/networking/sockets/examples/KKMultiServerThread.java`
- KKMultiServer.java" `https://docs.oracle.com/javase/tutorial/networking/sockets/examples/KKMultiServe`
  `java`

5 Policy

- You should develop your application in Java (or C/C++) programming language, in teams of two.
- You are required to submit external documentation for your program and to comment your code thoroughly and clearly. Your documentation should describe the overall organization of your programs, the major functions and data structures.
- No Late submission is accepted.
- You could not use a library (or a class) that provides the web server code. (If not sure, ask me)

6 Deliverables

- Report (10%)
  - Mention how you approach the problem.
  - Describe the classes you used, and document
  - them (you may include UML diagram)
  - User manual: how to run your program
- Code (70%)

  - Your code must compile with no errors
  - HTTP 1.0
  - HTTP 1.1 (10% bonus)
- Discussion (20%)

  You need to show that you understand the solution and the code.