Lab 2: A Web Server Model

Deadline: Octobor 13

Goal

Understanding HTTP and web server operation will give you a broad knowledge about the latest interconnection technologies and web services. In this lab, you will learn Socket network programming in C and understand the fundamental operations of web servers. Moreover, you will discover how web servers performance impact by the number number of requests.

Note: you must program this lab using C. I prefer to use Linux environment, where you can write shell scripts to run your code and/or analyze the server performance.

Tips and Guidelines

- At a fundamental level, the web server and the client are C programs. The server should accept HTTP request over a TCP network socket and the client should simulate a web browser.
- The web server should handle multiple incoming HTTP GET requests and send a response back to the client, with a proper Content-type in the response.
- You should test your code by running the client twice, one with a valid URL input and another with an invalid URL, use the terminal (no GUI is required). Furthermore, when the client receives a response from the server, it should print the response content in the terminal.
- Port 80 is reserved by the system for web servers, but your web server should listen and accept HTTP requests using a different port (e.g. 8080).
- When the web server receives a GET request from the client, it should parse the HTTP request and return an HTML file for the requested content or a suitable error message.
- If the server can't fulfill the client URL, it should send a proper status code, e.g., NOT FOUND Error 404, in the response.
- The most crucial part of the arrangement for this lab is to use at least two separate computers, one for the server and another for the client.

- You should use the HTML file (**Lab2.html**) to test your code. The HTML code embedded a GIF image, and both files can be downloaded from the lab section in the black board.

Building your Web server

The server should accept connections from a designated port, and this port should be assigned by default or as a user arguments before running the server. **For graduate students only:** another option needs to be available to the users where they can select the type of communication socket.

Here is an example below:

```
Undergraduate:
$> ./webserver <port number>
Graduate:
$> ./webserver <port number> <TCP or UDP>
```

The server should maintain a high availability and prevent from crashing.

Building your client

The client should parse the URL to get the server IP address and port number and establish a connection with the web server. It should accept one optional input; that is the requested URL. If the user didn't use the URL option then after a successful connection to the server, an inquiry in the terminal should appear and ask for the user to input a URL.

For graduate students only: another option needs to be available to the users where they can select the type of communication socket.

```
Undergraduate:
$> ./client [URL]
Graduate:
$> ./client [URL] <TCP or UDP>
```

If you are running both applications on the same machine, you can change the IP address to "hostname". The client should maintain a high availability and prevent from crashing. Additionally, the client needs to provide the user with meaningful reply if it fails to connect to the server.

Create the Project Repository

In LoboGit, create a new project in your group and name it **"Lab2"**. You should use this repository to maintain your code and collaborate with your group members.

Graduate Students

The requirements in this section intended for graduate students but undergraduates are welcome to do it for extra credit.

- Design one version of your client/server using TCP and UDP network sockets. You should select between the two implementations to perform the experiments (I don't accept two code versions one for TCP and another for UDP)
- Run experiments to simulate 100, 1000, and 10000 clients requesting the same web page from the server and measure the latency of requests. You should run the tests using TCP implementation and then repeat it using UDP. Draw a chart showing the average latency time between these experiments and link them to the used socket types. Moreover, your group should provide another chart that shows how many UDP requests received successful responses from the server in each trial. For this requirement, you should automate the process.
- As a graduate student, you should explain the outcome of the results and provide reasonable conclusions.

Submission

You have to solve this assignment in groups of two to four students. You must use a git repository (LoboGit), Lab 2 project, to maintain and submit your code. You can't change your group until you get approval from me, you should send me an email requesting such change with a good reason.

When you submit your work, the repository should contain the following:

- A wiki page, which includes a brief explanation on how your group solved the practices above with a walk through on how we can run your code. Be precise and brief in your writing.
- Your code WebServer.c, Client.c, and other C files. Make sure your code is well documented and readable, I will deduct points if the code misses documentation.
- **Extra credit:** an optional makefile to build your code.