CS422 Lab2: Build Your Own Web Server over Sockets

Due: 23:59:59 PM, Wed Feb 14, 2023
Total points: 75 points

1 Goal

This is a programming lab. In this lab, you are going to develop client-server socket programming and build a concurrent Web server in C. Also, we will take the chance to learn a little bit more about how Web browser and web server behind the scene.

2 Part A: Build Client-Server over Socket Programming (30 points)

- Set up server and client over Socket programming. The sample code and tutorial of socket programming are provided along with this assignment and lab instructions. Basically, the server opens a server process over TCP while each client creates a client process to connect to this server process. After you open the server, you can open the client and you will be prompted to input a message on the client, and then the message will be sent to the server after clicking "Enter". Upon receiving the message, the server will send the message back, which will be displayed on the client terminal. The server will record and display the message on its terminal, too.

- (5 points) You need to slightly revise the provided code to allow the client and server to run on different machines which can be specified by the IP address and port number in the command line. You should run the code using the following commands:

  ```
  ./server [port#]
  ./client [ip_server_addr] [port#]
  ```

  We expect that the client allows you to input at least ONE MESSAGE as the REQUEST. The server should print the received [REQUEST] in the terminal and capitalized the [REQUEST], and finally send back the capitalized [REQUEST] to the client. On the client side, both the original and capitalized [REQUEST] will be printed in the terminal when it is received by the client. You can find the sample output in sample_basic.txt

  Tip: The request may contain "\n". You should define the keyboard shortcut CTRL + D to signal that you have finished typing your request.

  You can begin by running both of server and client at the **localhost**. Namely, use *localhost* as [ip_server_addr] or simply use 127.0.0.1. After it works, you need to finally make the client and server run on two different machines (say, lab machines at HAAS G050), from amber01.cs.purdue.edu to amber30.cs.purdue.edu.

- (15 points) You need to revise the code to support multiple clients on the server using `Pthreads` and save it as `serverMul.c`. You can test the new server with at least two clients running on different machines or one machine with two different terminals. You run the code using the following commands:

  ```
  ./serverMul [port#]
  ./client [ip_server_addr] [port#]
  ./client [ip_server_addr] [port#]
  ```

We expect the server can connect with these clients at the same time and print REQUESTS sent by each client in the terminal. You can find the sample output in sample_mul.txt. Please follow the semantics of sample output files. Tips: To ensure the first client program does not terminate before you start the second client program, you can use “while” or for “loop” with “sleep()” in your code to repeat the client requests. You can also define your own way to close the client.

(10 points) Test your client code. Similar to page 36 in chapter 2 lecture slides, please use the client program you develop to try out HTTP. You need to first get your client to connect with a website (say, gaia.cs.umass.edu or other websites, port number: 80), and then try a correct and wrong GET request.

Please include the following results in your report:

1) a CORRECT GET request you used, the website tested, and the HTTP response the client received in the terminal. You don’t need to show BODY part but the status line and header lines.
2) a WRONG GET request you used, the website tested, and the HTTP response the client received in the terminal. You need to explain why this GET request is wrong and show the status line and header lines of HTTP responses.

You can find the sample output at sample_web_200.txt and sample_web_404.txt. Tip: please prepare your HTTP request message in advance (the same as I did in class) and then copy and paste the request message (not type it) in the terminal of the client.

3 Part B: Develop Your Web Server over HTTP/1.1(45 points)

Please read Chapter 2 of the textbook carefully. The textbook will help you to understand how HTTP works. Instead of using port 80 or 8080 or 6789 for the listening socket, you should pick your own to avoid conflicts. It is suggested not to use port numbers 0-1024.

This simple web server needs to implement the following features:

- When the client sends GET to request for one .html which exists, it should respond “*200 OK* ”and return this .html file.
- When the client sends GET to request for one .html which **doesn’t** exist, it should send the response “*404 Not found* ”.
- When the client sends GET with the wrong format (e.g. URL String Syntax Error), it should send the response “*400 Bad Request* ”.
- When the client sends GET with a different HTTP version, it should send the response “*505 HTTP Version Not Supported* ”.

Please save your server code as `server1.c`. If you also revise the client code to test all the above features and save your client code as `client1.c`.

Several files are provided to test with your web server. Of course, please feel free to create and use your own test files.

- text.html: a html file which contains text only.
- picture.html: a html file which contains text and a small picture (< 200KB).
- bigpicture.html a html file which contains text and a big picture (> 1MB).

You should run the code using the following commands:

```
./server1 [port#]
```

We expect the server can print the first line of client's (browser's) request and the first line of server's response (e.g., GET /index.php HTTP/1.1 $\backslash r \backslash n$ HTTP /1.1 200 OK). You can find the sample output at sample_http_response1.txt and sample_http_response2.txt.

Tip: You should send the response and the files to the client using the establishment socket.

You need to test your server with the browser:

- Create a folder named "www" in the folder containing your codes and programs. Include the relative path of "www" folder in your server's code and copy your .html files to it ONLY DURING TESTING. Please DO NOT INCLUDE "www" FOLDER OR ANY .html FILES IN YOU SUBMISSION. The same for Part C.
- (10 points) For your server, you should be able to see the HTTP request format in the console of your server machine. For your client, you should be able to show the specific response given different HTTP requests and the content of the requested file. In the following test, please copy the print on the console at the server side into your lab report.
- (20 points) Work with all three test files. Connect to your server from a browser with the URL of http://<machine name>:<port number>/<html file name> and see if it works. For example, you can try `http://localhost:8888/index.html` if your web server uses the port number of 8888 and has a file called index.html in its root folder for all the web contents. Please make sure the server should work with the existing browser (Chrome, Firefox, etc) to get the test webpages.

  Hint: the hard part is to handle large objects, say, bigpicture.html. Please do consider how to manage memory to transfer large objects.
- (5 points) Deploy your server in your local machine or any CS lab machine. Run tests to fetch three sample webpage files in the assignment. Please tell me the total time (round-trip-time) for the client to get each webpage file. (Hint: you can use wireshark or other tools provided web browsers to get the time needed. Please include your test results in the report, along with the approach you used.
- (10 points) Test with other HTTP responses (404, 400, and 505).

  Tip: Your server should be able to respond correctly in different cases. You do not need to make any change in client's code to get full credit for PART B, we only test your server1.

4 Materials to turn in

You will submit your assignment at gradescope. You submission should zip all the files into "lab2_PUID.zip", including the following files:

1. All your source code files (`serverMul.c`, `client.c`, `server1.c`). Please add appropriate comments to your code and make it easy to understand.
2. Your project report called `lab2.*` (txt, pdf or word format). In the report,
   - Please include your name and student ID at the top of the first page.
   - Include the answers and results as specified above.
   - Include a brief readme in the report to explain how to compile and run your source code, if needed. (if the grader can't compile/run your code, the project is considered not to be finished).

5 More Tips and Support

Please start early.

More questions about the assignment should be posted on Campuswire or asked during PSOs.