Programming Assignment 1

CSE 310 Spring 2024

Due date: **February 29, 2024; 11:59 PM**

Submission via Brightspace

Part A. (50 points) Web Server

In this part of the assignment, you will learn the basics of socket programming for TCP connections in Python: how to create a socket, bind it to a specific address and port, as well as send and receive a HTTP packet. You will also learn some basics of HTTP header format.

Develop a web server that handles one HTTP request at a time. Your web server should be able to (a) accept and parse the HTTP request, get the requested file from the server's file system, (b) create an HTTP response message consisting of the requested file preceded by header lines, and then (c) send the response directly to the client. (d) If the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client.

Running the Server

Put an HTML file (e.g., HelloWorld.html) in the same directory that the server is in. Run the server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26). You can also use localhost if the server is running on the same host. From another (or the same) host, open a browser and provide the corresponding URL. For example: http://128.238.251.26:6789/HelloWorld.html

'HelloWorld.html' is the name of the file you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number 6789. The browser should then display the contents of HelloWorld.html. If you omit ":6789", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80.

Then try to get a file that is not present at the server. You should get a "404 Not Found" message. You're not allowed to use a separate Html file for 404. It must be part of the HTTP response message.

Client should be able to request text as well as image (jpeg, png, etc) files from the server.

What to submit

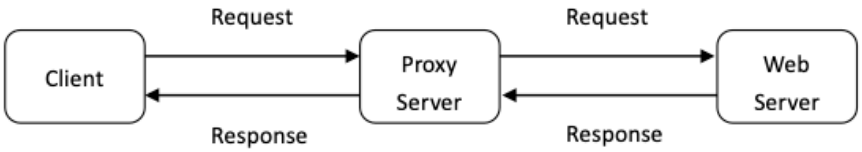
Please submit the complete server code (webserver.py) along with the screen shots of your client browser, verifying that you actually receive the contents of the HTML file from the server.

Part B. (50 points) Web Proxy

In this part of the assignment, you will learn how web proxy servers work and one of their basic functionalities – caching.

Your task is to develop a small web proxy server which is able to cache web pages. It is a very simple proxy server which only understands simple GET requests, but is able to handle all kinds of objects - not just HTML pages, but also images.

Generally, when the client makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client. In order to improve the performance we create a proxy server between the client and the web server. Now, both the request message sent by the client and the response message delivered by the web server pass through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client.



Running the Proxy Server

Run the proxy server program using your command prompt and then request a web page from your browser. Direct the requests to the proxy server using your IP address and port number.

For e.g. http://localhost:8888/www.google.com

To use the proxy server with browser and proxy on separate computers, you will need the IP address on which your proxy server is running. In this case, while running the proxy, you will have to replace the "localhost" with the IP address of the computer where the proxy server is running. Also note the port number used. You will replace the port number used here "8888" with the port number you have used in your server code at which your proxy server is listening.

Configuring your Browser

You can also directly configure your web browser to use your proxy. This depends on your browser. For example, in Internet Explorer, you can set the proxy in Tools > Internet Options > Connections tab > LAN Settings. You need to give the address of the proxy and the port number that you gave when you ran the proxy server. You should be able to run the proxy and the browser on the same computer without any problem. With this approach, to get a web page using the proxy server, you simply provide the URL of the page you want.

For e.g. http://www.google.com

What to submit

Please submit the complete proxy server code (proxyserver.py) and screenshots at the client side verifying that you indeed get the web page via the proxy server.

****

Submission Guidelines

You need to submit your homework in a single zip file as follows:

- The zip file and (the root folder inside) should be named using your last name, first name, and the assignment number, all separated by a dash ('-') e.g. lastname-firstname-assignment1.zip
- The zip file should contain your code corresponding to Part A and Part B. Please be sure to put code in the root folder rather than in separate folders. Provide sufficient comments in your code.
- Include the screenshots for Part A and Part B in a single PDF file, and name it 'clients_screenshots'.
- Directory structure you can follow:
  Files inside 'lastname-firstname-assignment1':
  - webserver.py
  - proxyserver.py
  - clients_screenshots.pdf
  - .html file for part A
  - README
- Flask and SimpleHTTPServer are NOT allowed. Please use socket programming only. You can use the server code discussed in class as a starting point.
- You cannot use urllib.parse library to parse URLs
- You can handle only HTTP requests. No need to handle HTTPS requests. Some websites will only support HTTPS. You may need to find one that uses HTTP.
- You can try the following test webpages, they should work with http1.0.
  - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
  - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html
  - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
  - http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file5.html
- You should provide a README file describing:
  - Any external libraries used.
  - Instructions on how to run your programs.
  - Webpages that your code successfully works for.