CS3640 Programming Project-1

Due: Mar 18, 2021 midnight
Upload to ICON a single zip file containing all materials

Part-A. Web Server **40 points**

The first part requires you to put into practice your understanding of socket programming. You will build a simple web server that is capable of handling one GET request at a time. Specifically, the web server will (i) create a connection socket when contacted by an http client, (ii) accept and parse the http request from that connection, (iii) get the requested file from the server's file system, (iv) create an http response message consisting of the requested file preceded by header lines, and finally (v) send the response back to the client. If the requested file is not present in the server, the server should send an http "*404 Not Found*" message to the client.

Code: We have provided a skeleton code (in Python) for the Web server at *https://shastri.info/teaching/cs3640/assignments/web-server.py*. The places where you need to fill in code are marked with `#FillInStart` and `#FillInEnd`. Each place may require one or more lines of code. If you prefer to use a language other than Python, please discuss with your TA before implementing your solution.

Running the server: Determine the IP address of the machine running the server, and select a non-standard port number for it. Then, run the web server after binding it to that port. Make sure to place a simple html file in the same directory as the server.

From another machine, open a web browser and type in the URL http://<server-ip-addr>:<server-port-number>/<file-name>. Make sure the web browser is able to display the file. Then, repeat the testing with a non-existent file. The browser should display "*404 Not Found*" message.

Submission: You should submit the complete server code along with the screenshots of your client browser showing the two file fetch scenarios.

Part-B. Web Proxy **60 points**

In the second part, you will build a web proxy server. It is a simple proxy which understands only http GET requests and is able to cache web pages. As we learnt in our class, a web proxy sits between a web client and a web server i.e., the client requests the objects via the proxy server. If the proxy server does not already contain that object, it will forward 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 (after caching in a copy for itself). Though in practice, the web proxy must verify that the cached response is still valid and that it is the correct response to the client's request, you can ignore these constrains for this exercise.

Code: We have provided a skeleton code (in Python) for the web proxy at *https://shastri.info/teaching/cs3640/assignments/web-proxy.py*. The places where you need to fill in code are marked with `#FillInStart` and `#FillInEnd`. Each place may require one or more lines of code. Again, if you prefer to use a language other than Python, please discuss with the TA before implementing your solution.

Running the server: The first step is to determine the IP address of the machine that would host the web proxy and select a non-standard port number for it. Then, run web proxy using your command prompt.

Once the web proxy is up, you will configure your web browser to use that proxy. It is preferred that you run your web browser on a different machine. Configuring the proxy is browser specific: if using *Internet Explorer*, you traverse Tools > Internet Options > Connections tab > LAN Settings; if it is *Chrome*, you navigate to Preferences > Settings > Advanced > System > proxy settings. In there, you will need to specify the IP-address and port number of the proxy server. If there is an option to choose only http/web proxy, please do so. If you use a different browser, and are unable to figure this out, please reach out to the TA.

Now, open your web browser and enter a http site (say, http://www.gnu.org/). Make sure the web browser is able to display the file. Then, reload the web page. This time, the file should have come directly from the web proxy. Make sure you validate this using web proxy logs. Finally, ask for a non-existent file (say, http://www.gnu.org/cs3640). Verify that receipt of “404 Not found.”

Submission: You should submit the complete web proxy code along with output from web proxy showing the three file access scenarios.