CSE5344 – Project 1 (Fall 2013)
Multithreaded Web Proxy Server

Objectives

- Gain exposure to socket programming
- Understand basic functionalities of a web proxy server
- Explore basic structures of HTTP GET and POST messages

Due: Nov 3, 2013, 11:59pm

Project Specification

In this project, you will implement a web proxy server. Your implementation should be able to perform the following functionalities:

- Receiving HTTP requests (GET and POST method) from a browser and forwarding them to the origin server.
- Sending corresponding HTTP responses receiving from the origin server to the client.
- Handling multiple requests at the same time (multithreaded proxy server).
- Handling errors when a client requests an object which is not available.
- Caching web pages each time the client makes a particular request for the first time and sending the cached web pages to the client when a cache hit occurs. You do not need to implement any replacement or validation of the cached web pages.

Please refer to socket programming assignment 4 of the textbook (Reference 3) for more detail on the functionalities of a web proxy server and the skeleton code in Python.

References

1. http://www.w3.org/Protocols/rfc2616/rfc2616.html: HTTP/1.1 specification
2. http://heather.cs.ucdavis.edu/~matloff/Python/PLN/FastLanePython.pdf: Introduction to Python
3. J. Kurose and K. Ross, "*Computer Networking: A Top-Down Approach*," 6th edition, chapter 2, section 2.2, 2.7: HTTP message format, Web caching, and Socket programming.

Notes

- An example syntax for running the proxy server:

```
proxy_server_code_name [port_number]
```

The optional argument 'port_number' is the port on which the proxy server is listening to a connection from a client. If the port number is not entered, the default port 8080 is used.

- For testing POST method, you can go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html and upload a file, e.g., *alice.txt*, to this website.

Submission Guidelines

- You will turn in the complete proxy server code and any additional programs as needed. In case your code is not working by deadline, send it anyway and review it with the TA for partial credit.
- The code must be well documented.
- You must submit a readme file in text format which includes instructions on how to compile and run your programs. The readme file must also mention the development environment used as well as any packages that might be required for running the codes.
- All source files and other necessary items must be zipped into a single file with the naming convention <your-uta-id>_<your-name>.zip which is emailed to the class TA (*mqnguyen@mavs.uta.edu*) with the subject "CSE5344 - Project 1". Do NOT include any runnable executable (binary) program.
- Make sure your name and your UTA ID are listed in the readme file and in the comments at the beginning of your source files.

Additional Requirements/Instructions

- Complete documentation and coherent instructions for running the code are recommended, otherwise you may be asked to come and give the TA a demo if he is not able to execute your programs from the instructions provided.
- If you are using any code from some external source or book, you must mention it explicitly in the code as well as the readme file. If not mentioned, it will be considered as plagiarism and your project will not be evaluated.
- You can discuss with other classmates on steps/algorithms to implement the project. However, the source codes must be written by yourself.

Grading (100 points)

- Receive/send messages from/to a web browser successfully (20 points)
- Receive/send messages from/to web servers successfully (20 points)
- Process GET method correctly (20 points)
- Process POST method correctly (10 points)
- Handle multiple requests at the same time (multithreaded implementation) (10 points)
- Handle errors when a client requests an object that is not available (5 points)
- Cache web pages properly (10 points)
- Well-documented codes (5 points)