Danny Stieben

Aaron Tabor

Ahmed Alotaibi

April 23, 2015

CSCE 3530

Project 2 Report

In Project 2, we were tasked with creating a web proxy server. The server would act as an intermediary between a web browser that connects to it and a web server that the browser would normally connect to directly. To provide some actual benefits of using the proxy, we were also tasked with adding various features to the proxy. These features included the ability to block access to certain websites, filter out bad words, and cache web pages so that repeated requests for the same page could be served from the cache rather than asking the website’s server respond again. Of course, it should also be able to handle multiple clients at the same time.

Our code is based off of the demo code that was provided to the class. Specifically, we used serverThreads.c as our starting point for the server. From there, the server was developed first to be able to simply facilitate communications between the browser and the website’s server while acting as the intermediary. Once the flow of communication was able to pass through the proxy server correctly, the other features were implemented to perform certain tasks with the information that was given to the proxy server from either the browser or the website’s server.

Then, the other features of the proxy server were added. To implement blocking of certain websites/domains, it compares the domain requested by the web browser to a list of blacklisted domains and sends back a “blocked” message if the domain is to be blocked. Otherwise, it fulfills the request.

To implement the bad word filtering, the proxy server captures the website’s server’s response, and then scans through that response for each bad word defined in the appropriate file. Once the response has been scanned and all defined bad words have been removed, the result is sent to the browser to be displayed.

Finally, to implement the caching of files, the proxy server first checks whether a file for the requested page (which has the same name as the address) exists. If it does, then it simply reads the contents of the file and sends it to the browser. If it does not, it captures the response from the website’s server and writes it into a new file that has the same name as the address.

Cached pages can be easily found in the same folder as where the proxy server is located, and there are also three configurable files responsible for the bad word list, the blocked domains list, and the response to send if the browser asks for a blocked domain.

The proxy server can perform the desired tasks with some added delays compared to a direct connection between a browser and web server. The proxy server is also not very reliable, as it tends to crash often, especially when the browser tries to issue multiple requests for essentially one webpage. However, it does work reliably enough that all of its features can be tested on relatively simple pages. Thanks to the multi-threaded architecture of the server, multiple clients can connect to the server at the same time and perform actions on it.

To run the code yourself, please refer to the README file for compilation and run instructions.

Individual Journals

Aaron:

April 18 – Helped plan the features of the proxy server

April 20 – Wrote portion of the report

April 22 – Attempted to look up a way to use regex for bad word filtering, but was unsuccessful

Ahmed:

April 21-22 – Worked on his own version of the proxy server, mainly by working on the additional features.

April 22 – Compared with Danny’s version to choose the best code/implementations

Danny:

April 20-21 – Worked on passthrough capability of proxy server

April 22 – Implemented extra features of proxy server, borrowed some portions from Ahmed’s code

April 23 – Additional testing, bug fixes and bug fix attempts.

All group members are satisfied with their share of work on the project, and we unanimously agree that all members should receive full participation points.