Sidharth Gilela
Programming Assignment 4
sgilela@ucsc.edu

Design Document

Goal:

The objective of this project was to better understand how a proxy works in receiving and sending information. It has allowed us to get a better understanding of how the communication between the browser, proxy, and web server all work with each other. We had to additionally implement concurrency on the server so it was great getting to further gain experience in using threads because that is the approach I chose for concurrency. By now, I can very conveniently use threads and make the most of use of them. It was great learning about the HTTP protocol as well the much complexity going on behind the scenes.

Compile:

To compile the proxy from src folder:

```
make
```

Design:

The design of my protocol will be explained by a sample run through my code. The first thing I do is some basic error checking like making sure that the file with the websites exist and the number of arguments don’t exceed 3. Afterwards, I create a thread pool with 256 threads. All of these threads get created and then they all get joined. Inside the thread’s method, I first call accept. This means that all my threads are waiting at accept. After a thread completes it’s execution, it goes back to the top and waits at accept due to an infinite while loop. This way, I can handle as many requests as the browser would like and am not limited to only 256 connections. Inside the thread’s method, I first do a receive in order to to retrieve the request sent by the browser. After doing this receive, I parse the buffer to add in the forwarded header. Thus now, I have a request from the browser that contains the Forwarded header. After this I check the length of the received buffer to make sure it doesn’t surpass the max limit of bytes a request could have. If the length does exceed the maximum, an error message of 413 is sent back to the browser. If this does not occur, I then parse through the request to find the type of request like GET or HEAD or neither. Depending on the type of request I do certain things. However, if the request was HEAD or GET, I implement both these requests the same way. If it was neither of these, an error message of 501 is sent back to the browser telling the browser that the method is not implemented. Once I get this HEAD or GET request, I further parse the request to get the website url which is next to the HOST field in the buffer. I then check if this website exists in the forbidden list of websites. If it is, a 403 error is sent back to the browser telling the browser that the website is forbidden. If this is not the case, I then check if this website has an IP address or not by doing the getaddrinfo system call. If there is no IP address with the website then a 404 error of Not Found is sent back to the browser telling the browser that the website cannot be found. If this is not the case, I then create a socket on the proxy to communicate with the web server. If this socket is not able to connect with the web server, then an error message of 502 is sent back to the browser so that way the browser is notified. When this happens, the new socket created is closed. If this does not happen and the proxy is successfully able to connect with the web server, then it send the buffer over to the server and waits for a response. It will wait up to 8 seconds for a response. If there is no response from the server within 8 seconds, an error message of 504 is sent back to the browser. If the proxy is able to get a response from the server, it will send that information directly back to the browser by using the socket file descriptor returned by accept. Unfortunately, at this place I was not able to do a continuous while loop to send the data from the proxy to the browser that the server sends to the proxy. When I do this I get a 501 error, so instead I can only send back the buffer once to the browser. It will send 9216 bytes back to the browser. The browser then presents the response from the server on to the screen so we are able to see the output. I have this access.log that records any requests from the browser to the proxy. When running the program again, you do not need to delete the access.log because I have written code that takes care of that. Referring back to the error messages, the error messages that I have sent back to the browser are custom made so they have the content length of the custom messages I have created. All of this together, is the summary of my protocol that I have created.