Harshita Shyam
4404640

2nd May, 2014

Assignment 7: Schematic Design of my Proxy Server

The Proxy server consist of two classes:
1. Main.java
2. ProxyServer.java

Main.java:
The Main class reads the configuration (In order to avoid conflicts when each thread will read them), init socket with client (browser), init logger and creates ProxyServer threads. When you run the application, you need to specify in console as parameters port, on which the application will work and the path to the configuration file (at the moment its path – assignment7/proxy_server/config.txt). In order to record the information on application performance, java.util.logging.Logger is used and because it is a synchronous, you can write logs from multiple threads simultaneously.

ProxyServer.java:
ProxyServer organizes basic work of the proxy server for interaction between the browser and between the original server used sockets. The proxy server acts like a HTTP server to the HTTP client and as HTTP client for a HTTP server. The HTTP client connects to the proxy server instead of directly connecting to the HTTP server. In order for that, the client connects to the proxy server, the proxy server keeps on listening on a particular port. Once the proxy server gets a request from the client, it opens a connection to the HTTP server mentioned in the request. The proxy server then sends the HTTP request to the HTTP server on this connection. The proxy server waits for the HTTP server to respond. When the proxy server receives the HTTP response, it forwards that response to the HTTP client. The Two main components here are:

1. only-if-cached - the server never sends a response from the origin server, first response is recorded in the cache, and then read from the cache. That is, it writes a file in the cache and then sends it to the browser. All the requests are processed as only-if-cached because I cannot force the browser to always sent requests as only-if-cached.

2. max-age - every time when accessing a file from cache, the validity time is verified , and if the file is obsolete, the server updates its (replacing the old file to the new one). Therefore, max-age is used to see how many seconds cache can store this file. Also, if-modified-since header is used to know how much time has

passed since the file was last modified. If (current-time - if-modified-since) > max-age then, we update the file.

The principles of server operation :

When the proxy receives a request from a browser, it opens input and output streams from socket, read request and stores it in the array of bytes, in order then to send the request to the original server, if needed. Further, the proxy parses request headers, and store them in HashMap. Then the proxy checks the type of operation and generates a path to the file, depending on a URL, that can be stored in the cache. Since we need to emulate the browser only-if-cached requests, proxy always tries to find the cached file to send it to the browser.

Path to the file in the cache is composed as follows:
All cached files are stored in CacheFolder directory. The file name is composed of the following parts :

1. Server host name (for example www.gnu.org)
2. Double underline
3. Relative path to the file from request url, where all '/' symbols are replaced with '-'.

CacheFolder directory has no internal folders, and consists of files with names, for example, www.gnu.org__feed-icon-10x10.png (original URL - www.gnu.org/feed/icon/10x10.png).

Program Execution :

- If website is blocked, proxy sends error to the browser.
- If website is not blocked, proxy is trying to find a file in the cache.
- If file not found – proxy sends request to original server and stores in cache it response. Then file from cache sent to the browser.
- If file exists – proxy check its max-age header. To test the max-age proxy reads If-Modified-Since to determine the time, which have elapsed since the file was last modified.
- If more time than specified in max-age header – proxy updates the file in the cache downloading a new version of the file from the original server. Else – proxy returns to the browser file from cache.
- When proxy gets the response from the original server it checks the file mime type. If the server name and mime type blocked – proxy removes stored file from cache and sends error to the browser.

To run the program:

javac ProxyServer.java
javac Main.java

java Main config.txt port number (right now it is 50024 on remote04.cselabs.umn.edu)

Then open in the browser (for example: www.yahoo.com). For the first time you can see in console that proxy get files from original server. But if you refresh web page in browser, you can see that proxy gets files from cache. Also you can see this log information in log file, called “application_log.txt”.

Note: When I tested it, it didn’t store blocked content in the cache but it still responds with it in the server. Also, the logs that are seen in terminal are stored in “application_log.txt” and the other logs are mentioned in the assignment pdf are stored in files in CacheFolder. The file logfile remains empty.