Lab #2: A Simple Web Proxy Client

Jahir Sadik Monon **Date: 04.02.2022**

Roll: 32

Program Documentation

I've implemented the web client using python language. No higher-level python libraries like requests, urllib etc. were used to implement the code. The program opens the URL given in the command line argument in the browser & also prints the HTML response object in the terminal. It implements a local cache that saves previously requested HTML files and only updates the file using Conditional GET statement if the Last-Modified date is updated. The program logic is as follows:

1. Take terminal URL input from the user and parse protocol, hostname, and path from input URL.
2. Load local saved cache info into a python dictionary, create cache related files if they don't exist.
3. If no file associated with input URL exists in local cache, check the status of the server by making a request using the HEADER method. If the server is live (status code: 200), fetch the requested HTML object.
    a. Check the fetched objects header, if it has a last-modified date in its header, save the HTML file, also the Last-Modified date along with the associated URL in local storage. Open in browser and print HTTP response in terminal.
    b. If there's no Last-Modifier field in the fetched HTML files header, don't save it in local storage, just print HTML response and directly open link in browser.
4. If the file associated with input URL exists in local cache, check the status of the server by making a request using the HEADER method. If the server is live (status code: 200), Send a Conditional GET request to the server to check if the cached file is up-to-date.
    a. If the cached file is up-to-date (status code: 304), print the HTTP response for Conditional GET and open the link using the HTML file saved in local storage.
    b. If the cached file isn't up-to-date, print the HTTP response, directly open the link in browser, and update the cache information.
5. Close the program after printing on terminal and opening the URL in the browser.

In the program, the parse_url_fields(...) method parses the URL fields, the fetch(...) method uses the socket library to fetch the binary data from the host and converts it to a string, the cache_and_open(...) method is used to cache the file and open the .HTML file in the browser.

Assumptions

1. Filenames can be generated from the hash output of their hostname and path info, which should be unique.
2. We have to first test if the server is live, using HEAD method and its response status code.
3. We are caching a file only if the Last-Modified date is specified in its response, as otherwise we cannot use the Conditional GET later on.
4. We have to implement the program without the use of higher level python libraries.
5. Lastly, we've assumed that this program won't be used for URLs that require sophisticated browser headers and other certificates.