CSE 4074 – Programming Assignment

Due 29.12.2024 Sunday, 11:59 PM

Socket Programming – HTTP Server and Proxy Server

In this project, you are required to implement a multi-threaded HTTP server that returns a document determined by the requested URI. Additionally, you will develop a proxy server with specific functionalities. The performance of your servers will be evaluated using the "wrk" testing tool.

1) Implementing a multithreaded web server

You are required to implement an HTTP server that achieve the following requirements:

a) Your server should be capable of providing concurrency via multi-threading.

b) Your server program should take single argument which specifies the port number.

c) Your server should return an HTML document according to the requested URI. The size of the document is determined by the requested URI (any size between 100 and 20,000 bytes). Your server should remove the leading slash '/' from the URI and extract the resulting string as an integer (in decimal) that specify the size of the document to be sent to the client. For example, if the request line is "GET /1500 HTTP/1.0", your server should send back an HTML file that contains exactly 1,500 bytes of text (When the client save the document on the local disk, the file size should be 1,500 bytes). The returned HTML file should have a proper HTML format with <HTML>, <HEAD> and <BODY> tags, but the content can be anything. If the requested URI asks for a size less than 100, or for a size greater than 20,000, or if the URL is not an integer, your server should not return any document, but instead it should return a "Bad Request" message with error code 400. If the method in request message is not GET, the server would return "Not Implemented" (501) for valid HTTP methods other than GET, or "Bad Request" (400) for invalid methods.

1500 ü al parsla sonra int çevirip client a yolla

As an example, the following document is 100 bytes long:

```
<HTML>
<HEAD>
<TITLE>I am 100 bytes long</TITLE>
</HEAD>
<BODY> a a a a a a a a </BODY>
</HTML>
```

d) Your server must send back an HTTP response line, a Content-Type header and a Content-Length header. None of these count towards the size of the document.

e) Your server should send back an HTTP response line that indicates an error if the requested URI is not a number, or is less than 100, or is greater than 20,000.

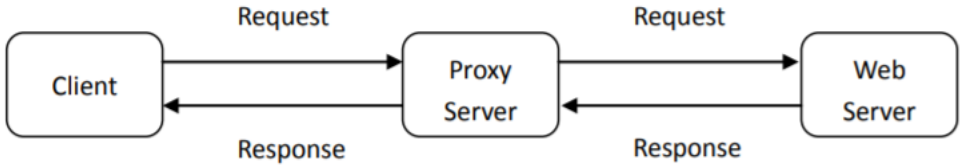
f) Your server should print out information about every message received and every message sent.

g) Your server should work when connected via a web browser (such as Internet Explorer or Google Chrome). For example, if the server port number is 8080, http://localhost:8080/500 would be a valid URL if the server runs in the same host..

2) Implementing a proxy server

You are required to implement a proxy server that achieves the following requirements:

a) Your proxy server will not be able to cache HTTP objects. It just relays the request to the Web server implemented in the first step and send back the result to the client (as shown in the figure below).



b) Proxy Server should use port 8888. Any client should be able to send a GET message to the proxy server as described in 1-c. But the proxy server would not generate any file, it would just direct GET message to the Web server. The result received from the web server would then passed to the client.

c) Your proxy server only directs the requests to your web server. It doesn't direct any request to another web server. **(+3 pts bonus if your proxy directs to any web server)**

d) In this project, proxy server has a restriction. If the requested file size is greater than 9,999 (in other words, if the URI is greater than 9,999) it would not pass the request to the web server. Rather it sends "Request-URI Too Long" message with error code 414.

e) If the Web Server is not running currently, your proxy server would return a "Not Found" error message with status code 404.

f) The proxy server should work when connected via a browser after configuring the proxy settings of your browser. (Please check how to configure your web browser to use proxy. For example, in Internet Explorer, you can set the proxy in Tools > Internet Options > Connections Tab > LAN Settings). Enter IP address of your web server for the proxy address (127.0.0.1 for localhost) and 8888 for the port.

When you configure proxy settings, all requests would be first directed to the proxy server. Your proxy server will accept a valid GET message from the client and forward it to the web server as follows:

Accept from client:
GET http://localhost:8080/500 HTTP/1.0

Send to web server:
GET /500 HTTP/1.0
Host: localhost:8080
(Additional client specified headers, if any…)

Note that it accepts absolute URI (http://localhost:8080/500) from the client, while it sends relative URL (/500) to the server together with a Host header. According to RFC 2068 section 5.1.2, "The absoluteURI form is required when the request is being made to a proxy". In addition, if the client requests a relative URL (such as GET /500 HTTP/1.0) from the proxy, proxy will direct this request to your web server as default.

Bonus: +15 pts if your proxy also provides caching. The proxy server will cache all the requested objects in the file system and if the objects are requested again, it will not forward these requests to the origin server. The proxy server will accept the cache size as an input. You can implement any replacement policy when the cache reaches its capacity (such as First-in-First-Out (FIFO), Least Recently Used (LRU)). (+5 pts will be awarded

for implementing functionality related to Conditional GET. Assume that files with an even length are always modified, while files with an odd length remain unmodified on the server. However, the proxy is unaware of this behavior.)

Note: If both your proxy server and web server run in localhost (127.1.1.1), then you may experience problem while reaching your server via your proxy. (Your browser may bypass your proxy server even if you set your proxy address and port number). If the server is in another host, this is not a problem.

In Windows, you can force visiting the proxy first (for a server running in localhost) as follows:

- Open the OS's proxy setting
- Ensure your proxy server (host & port) are set.

- Below, there will be an input which says "Use the proxy server except for address that start with the following entries. Use semicolons(;) to separate entries"
- Add <-loopback> into that input.
- Save.

For Linux, just add a new entry to your "/etc/hosts" file:
127.0.0.1 local
(This solution probably works for macOS as well)

Note that, after setting your proxy server, you will write the same address to the URL. Your browser will directly send an absolute URL to your proxy. You will not write address/port of the proxy server in the URL.

3) Testing your server using wrk
Use wrk for testing your web server.
Please refer to https://github.com/wg/wrk for details related to wrk.
Design your own **stress test** for testing the performance of your server. Provide the appropriate performance metrics (such as average latency, maximum latency, request per second, etc) for various number of threads (-t argument of wrk) and various number of parallel connections (-c argument of wrk). Try to answer the following questions:

- How your server's performance change by increasing level of concurrency.
- Maximum of how many concurrent requests do your server handle in a reasonable time?

Comment on your results.

Repeat same tests for the proxy server. Comment on your results.
It is recommended to use wrk; however, for the Windows operating system, you may alternatively use winrk. (https://github.com/fomalhaut88/winrk)

You can do your project in groups of **three**. But you have to give the names of your group members before December 9, Monday.

What to submit? - You should submit your projects in a zip file which contains your well COMMENTED source code and DETAILED report via **google classroom**.
Name of the zip file should be: name1surname1_name2surname2_name3surname3.zip. No need to write your IDs on the file names. However in your report, clearly indicate student IDs and names of group members. Detailed report should include design document, implementation details and answers to the above questions.