

## CSE 474 – Programming Assignment

Due 16.12.2018 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. You are also requested to implement a proxy server with some specific properties.

You are also required to use ApacheBench program for testing your servers.

#### 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, your server should not return any document, but instead it should return a "bad request" message with error code 400.

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

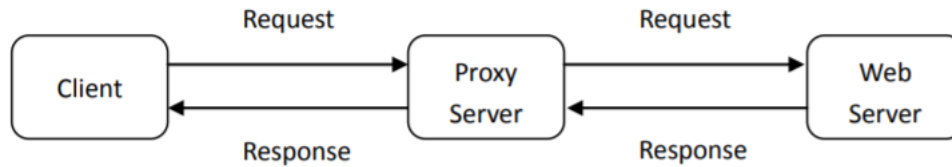
```
<HTML>
<HEAD>
<TITLE>I am 100 bytes long</TITLE>
</HEAD>
<BODY> a a a a a a a a </BODY>
</HTML>
```

- 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.

## 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 be 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.

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. You can directly use the URL (for example, if the proxy server runs in the local host, then <http://localhost:8888/500> would be a valid URL). Or you should be able to configure 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). When you configure proxy settings, all the request would be first directed to the proxy server.

**3) Using ApacheBench (ab) program:** This program can generate multiple HTTP requests to a web server with any desired level of concurrency. Below is a sample usage of the ab program to give you an idea of what it does. The test run below sends 100 requests to mimoza.marmara.edu.tr for the URI ~omer.korcak, sending 10 requests at a time (10 clients at a time).

```
$ ab -n 100 -c 10 mimoza.marmara.edu.tr/~omer.korcak
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking mimoza.marmara.edu.tr (be patient).....done

Server Software:      Apache
Server Hostname:      mimoza.marmara.edu.tr
Server Port:          80

Document Path:        /~omer.korcak
Document Length:      250 bytes

Concurrency Level:    10
Time taken for tests:  2.905 seconds
Complete requests:    100
Failed requests:       0
Non-2xx responses:    100
Total transferred:    49900 bytes
```

```

HTML transferred:      25000 bytes
Requests per second:   34.42 [#/sec] (mean)
Time per request:      290.539 [ms] (mean)
Time per request:      29.054 [ms] (mean, across all concurrent requests)
Transfer rate:         16.77 [Kbytes/sec] received

```

#### Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	16	28 7.1	31	42
Processing:	45	242 45.1	254	285
Waiting:	31	174 61.6	180	277
Total:	71	271 45.4	285	316

#### Percentage of the requests served within a certain time (ms)

50%	285
66%	285
75%	287
80%	290
90%	301
95%	301
98%	306
99%	316
100%	316 (longest request)

You are required to download and run ab in your own PCs. You can use ab in either Windows or Linux (if you have any trouble to download and run ab, please contact to instructor). You can get detailed usage information by typing 'ab -h'.

Using ab program, send 10 requests to <http://ipv4.download.thinkbroadband.com/5MB.zip> using following parameters:

- Send single request at a time (concurrency level = 1)  
(use "ab -n 10 -c 1 <http://ipv4.download.thinkbroadband.com/5MB.zip>").
- Send 5 requests at a time (concurrency level = 5).
- Send 10 requests at a time (concurrency level = 10).
- Repeat a), b) and c) using -k argument. This argument will send a "Connection: Keep-Alive" header to the web server.

For each of the above cases, give the following obtained outputs. (It would be preferable to repeat the above tests several times and get the average).

- Time taken for tests (seconds)
- Total transferred (bytes) and HTML transferred (bytes)
- Time per request (ms)
- Requests per second (#/sec)
- Transfer rate (Kbytes/sec)
- Connection times

Clearly describe how and why these values change when you change concurrency level and when you use -k argument.

### 3) Testing your server using ApacheBench

Use ApacheBench for debugging and testing your web server. "HTML transferred" should fit the requested document size. Moreover, perform a **stress test** for testing the performance of your server, as follows. Starting from one, increase number of parallel requests one by one and obtain the "time per request" and "requests per second" values. How do these values change by

increasing level of concurrency? Maximum of how many concurrent requests do your server handle in a reasonable time? Comment on the results.

Repeat same tests for the proxy server. Comment on the results.

You can do your project in groups of **two**.

**What to submit?** - You should submit your projects in a zip file which contains your well COMMENTED source code and DETAILED report by e-mail to [cse474submit@gmail.com](mailto:cse474submit@gmail.com).

Please do not forget to write the names of people in your groups. Detailed report should include design document, implementation details and answers to the above questions.