# Assignment 2

## HTTP Proxy Server (15 marks)

### Problem (6 marks)

Your task is to code a small HTTP web proxy server which is able to cache web pages. It is a very simple proxy server which only understands simple GET requests, of the form:

```
GET    /index.html    HTTP/1.1
```

Generally, when the client (browser or curl command) makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client.

In order to improve the performance we create a proxy server between the client and the web server. A web proxy is a program that acts as an intermediary between a web client (browser or curl) and a web server.

Now, both the request message sent by the client and the response message delivered by the web server pass through the proxy server.

In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client.

### Setting Up Server

We will use this server instead of any outside server, to request the files from.

Download this server's code from moodle and run the script "server.py". This will launch a server that serves files from its directory. We will use this server to get the files from. It runs on port 20000.

Test the server by requesting the URL *http://127.0.0.1:20000/file_1.html* from the browser. to get the *file_1.html* file from the home directory served at *http://127.0.0.1:20000*.

It can be considered equivalent to requesting this URL *http://www.something.com/pdfs/ebooks/abc.pdf* from the browser and getting *abc.pdf* file from *pdfs/ebooks/* directory served on *http://www.something.com* server.

## Running the Proxy Server

Run the proxy server script which you wrote on the terminal. Assume you are running your proxy server on localhost (127.0.0.1) and port number used is 12345.

Request the webpage or a file from your above running server through either browser **or** curl.

Using curl

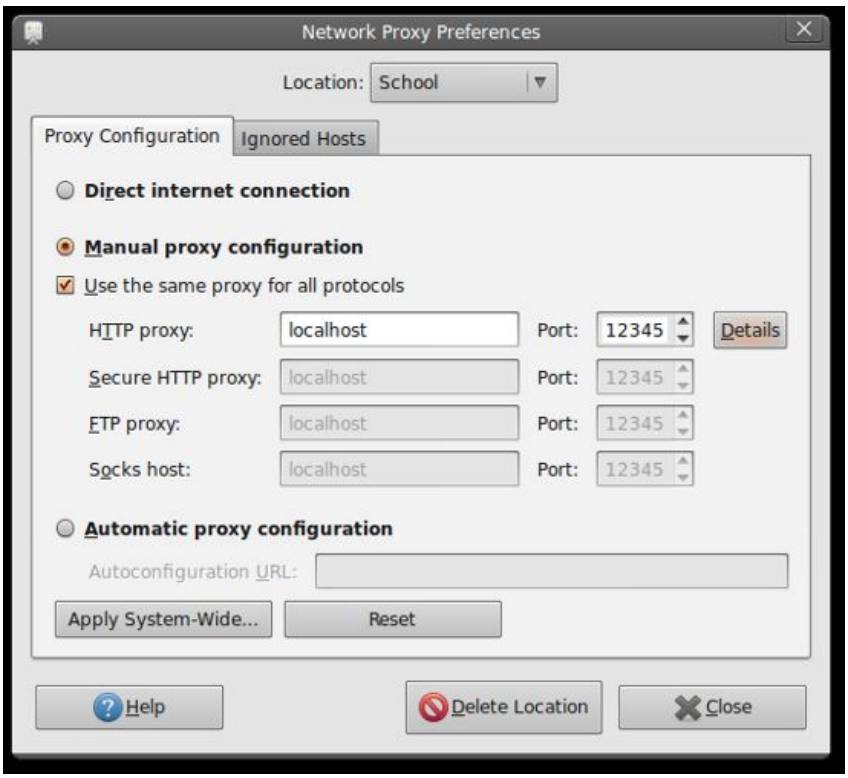
curl -x protocol://host:port *http://www.something.com/pdfs/ebooks/abc.pdf*

In this case, it is:

curl -x http://localhost:12345 *http://127.0.0.1:20000/file_1.html*

Using browser

For browser to use the proxy, you'll need to set the proxy by changing in your system's network settings in case of Chrome or macOS and browser's network settings in case of Firefox. You need to give the host and the port number where your proxy server is running, as follows:



You should be able to run the proxy and the browser on the same computer without any problem. With this approach, to get a webpage or a file using the proxy server, you simply provide the URL of the page or file you want.

## Caching (5 marks)

When the proxy server gets a request, it checks if the requested object is cached (i.e. server already has the request webpage or file), and if yes, it returns the object from the cache, without contacting the server.

If the object is not cached, the proxy retrieves the object from the server, returns it to you and caches a copy of this webpage for future requests. (More details here)

In case of any further requests for the same, the proxy must utilize the “If Modified Since” header to check if any updates have been made, and if not, then serve the response from the cache.

In practice, the proxy server must verify that the cached responses are still valid (that is, haven't been updated) and that they are the correct responses to the client's requests.

Note:

1. You can keep the data structure to track if certain page is there in the cache and the actual content of the webpages in your main memory, no issues. In practice, the webpage content is written to/read from the disk.
2. You can keep up to only 3 responses in the cache.
3. Use If Modified Since header. Read about it here.

### Bonus (1)

Create a non blocking server, that is, multiple clients can send requests at once. Post on moodle to get more details.

### Viva(4)

### Instructions

1. You only need to handle HTTP GET requests.
2. Note that all HTTP requests cannot be cached, and is dependant on the Cache-Control header.
3. The languages permitted for this assignment are C, C++ and Python.
4. Use HTTP status codes, like 404, 200, 304 issued by server. You can check the server.py file provided for what all response codes it can return.

5. All error scenarios must be gracefully handled. Programs crashing during testing will be penalised.
6. Plagiarism in any form shall not be tolerated.
7. Bonus will be used to make up for lost marks in other parts. Maximum marks of assignment is 10.

**Submission Format**

In your submission, provide the

- complete code for your proxy server
- a screenshot of your browser or curl command showing the contents of webpage or other file
- a README

Put all the files together in a <roll number1_rollnumber2>.tar.gz to submit them on Moodle.