Project 4: HTTP Web Proxy Server

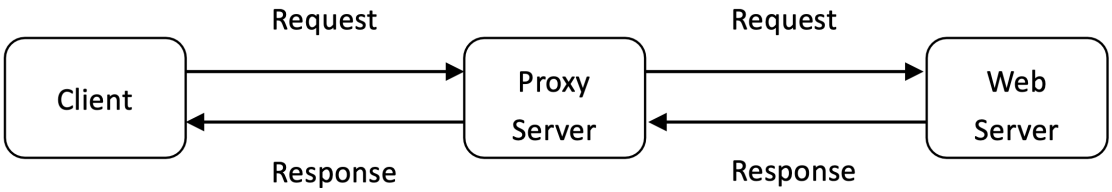
Due: 11:59pm, March 3 (Friday), 2023

(total points: 100)

In this project, you will learn how web proxy servers work and one of their basic functionalities — caching.

Your task is to develop a small web proxy server that is able to cache web pages. It is a very simple proxy server which only understands simple GET-requests, but is able to handle all kinds of objects – not just HTML pages, but also images.

Generally, when the client 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. Now, both the request message sent by the client and the response message delivered by the web server passes 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.



Below you will find the skeleton code for the mail client. You are to complete the skeleton code. The places where you need to fill in code are marked with #Fill in start and #Fill in end. Each place may require one or more lines of code.

Running the Proxy Server:[1]

Run the proxy server program using your command prompt and then request a web page from your browser. Direct the requests to the proxy server using your IP address and port number. For example:

```
http://localhost:8888/www.youtube.com
```

To use the proxy server with browser and proxy on separate computers, you will need the IP address on which your proxy server is running. In this case, while running the proxy, you will have

[1] You can also directly configure your web browser to use your proxy. This depends on your browser. In Internet Explorer, you can set the proxy in Tools > Internet Options > Connections tab > LAN Settings. In Mozilla, you can set the proxy in Tools > Options > Advanced tab > Network tab > Connection Settings. In both cases you need to give the address of the proxy and the port number that you gave when you ran the proxy server. You should be able to run the proxy and the browser on the same computer without any problem. With this approach, to get a web page using the proxy server, you simply provide the URL of the page you want. For example: www.youtube.com.

to replace the “localhost” with the IP address of the computer where the proxy server is running. Also note the port number used. You will replace the port number used here “8888” with the port number you have used in your server code at which your proxy server is listening.

Caching: A typical proxy server will cache the web pages each time the client makes a particular request for the first time. The basic functionality of caching works as follows. When the proxy gets a request, it checks if the requested object is cached, 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 the client and caches a copy for future requests. In practice, the proxy server must verify that the cached responses are still valid and that they are the correct responses to the client’s requests. You can read more about caching and how it is handled in HTTP in RFC 2068. Add the simple caching functionality described above. You do not need to implement any replacement or validation policies. Your implementation, however, will need to be able to write responses to the disk (i.e., the cache) and fetch them from the disk when you get a cache hit.

```
from socket import *
import sys
import ssl

if len(sys.argv) <= 1:
    print('Usage : "python ProxyServer.py server_ip"\n[server_ip : It is the
    IP Address Of Proxy Server')
    sys.exit(2)

# Create a server socket, bind it to a port and start listening
tcpSerSock = socket(AF_INET, SOCK_STREAM)
# Fill in start.
# Fill in end.

while 1:
    # Strat receiving data from the client
    print('Ready to serve...')
    tcpCliSock, addr = tcpSerSock.accept()
    print('Received a connection from:', addr)
    message = # Fill in start.          # Fill in end.
    print(message)
    # Extract the filename from the given message
    filename = message.split()[1].partition("/")[2]
    print(filename)
    fileExist = "false"
    filetouse = "/" + filename
    print(filetouse)
    try:
        # Check wether the file exist in the cache
        f = open(filetouse[1:], "r")
        outputdata = f.readlines()
        fileExist = "true"
        # ProxyServer finds a cache hit and generates a response message
        tcpCliSock.send("HTTP/1.0 200 OK\r\n")
        tcpCliSock.send("Content-Type:text/html\r\n")
        # Fill in start.
        # Fill in end.
```

```
            print('Read from cache')
    # Error handling for file not found in cache
    except IOError:
        if fileExist == "false":
            # Create a socket on the proxyserver
            cc = # Fill in start.    # Fill in end.
            hostn = filename.replace("www.","",1)
            print(hostn)
            try:
                # Connect to the socket to port 443
                context = ssl.SSLContext(ssl.PROTOCOL_TLSv1)
                c = context.wrap_socket(cc, server_hostname=hostn)
                # Fill in start.
                # Fill in end.
                # Create a temporary file on this socket and ask port 80
                # for the file requested by the client
                fileobj = c.makefile('r', 0)
                fileobj.write("GET "+"http://" + filename + "HTTP/1.0\n\n")
                # Read the response into buffer
                # Fill in start.
                # Fill in end.
                # Create a new file in the cache for the requested file.
                # Also send the response in the buffer to client socket and
                # the corresponding file in the cache
                tmpFile = open("./" + filename,"wb")
                # Fill in start.
                # Fill in end.
            except:
                print("Illegal request")
        else:
            # HTTP response message for file not found
            # Fill in start.
            # Fill in end.
     # Close the client and the server sockets
    tcpCliSock.close()
# Fill in start.
# Fill in end.
```

Question 1: [100 points]

Complete the above code to implement the proxy and caching functions. Due to the simplicity of the security implementation, your proxy server may not support some web access (e.g., `http://localhost:8888/www.cnn.com` and `http://localhost:8888/www.google.com`). But it should support the following three accesses. [I just tested them. They work!]

`http://localhost:8888/www.youtube.com`

`http://localhost:8888/www.bbc.com`

`http://localhost:8888/www.sohu.com`

In the above link, replace `localhost` with your proxy server's IP and replace 8888 with your

proxy server's port number.

What to Hand in

You are to submit two files: (1) Your completed Python code, and (2) a report including your screenshots and detailing your observations/explanation. For the report, please use the provided template.