CS305 Computer Network

Lab Assignment 1

Name: 李佳骏

SID: 11912021

Design idea

The overall idea is to create a proxy server that receives and sends requests and responses.

Connecting to the client

Firstly we create a server socket and bind it to our PC's IP and port, named *tcpSerSock*, which is used for listening and accepting the connection request with the client.

After connected with a client, we create a new socket named *tcpCliSock*, which is for processing requests and responses with the client. Then the proxy socket receives "GET", "HEAD" or "POST" requests, and creates a filename with the url found in the request message. The filename will be later used in the caching procedure.

Caching with Modification Notice

The caching design involves error handling. "try....except" structure is applied in the design.

First we try to read the design in the cache, if there is no requesting records of a website in the cache, an IOError would occur, which would be handled and a new file named after the url will be created in the memory, and the response message will be stored in the file.

If there exists such file in the cache, we inspect the start line of the response to see what the method is. If it is "GET" or "HEAD", we will add an extra *If-Modified-Since: GMT Time* line to the request message. The GMT time is the time found in the *Last-Modified* line from the cached response. Then we decide whether to update the cached file based on the state number from the new response.

GET, HEAD and POST Requests

The basic idea is to receive these requests from the client and send them to the host server, then read the respoonse from the server.

In the "connecting to the client" phase, we have created *tcpCliSock* to connect to the client. Now we create a new Socket named *tcpHostSock* to connect to the host server.

If an IOError occur, we use the *tcpHostSock* to send the request message and receive the response. Then we send the reponse to the client via *tcpCliSock*.

If the file can be found in the cache, if the request uses "POST" method, we send the response message in the file to the client directly, and if the request uses "GET" or "HEAD" methods, we apply the *tcpHostSock* to send

the modified request message and receive the response. Then the cached content will be sent to the client via *tcpCliSock*.

Multithreading

The package **threading** is imported to support multithreading. The procedure is as follows.

```
from socket import *
import threading

# default IP and PORT
IP = '127.0.0.1'
PORT = 8080

def thread(ip, port):
    # Main body of the code
    #...
    #tcpCliSock.close()

t = threading.Thread(target=thread, args=(IP, PORT))
t.start()
```

Testing

Proxy GET and HEAD Request

1. GET

```
C:\Users\10098>curl -x 127.0.0.1:8080 http://www.example.com/
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

2. HEAD

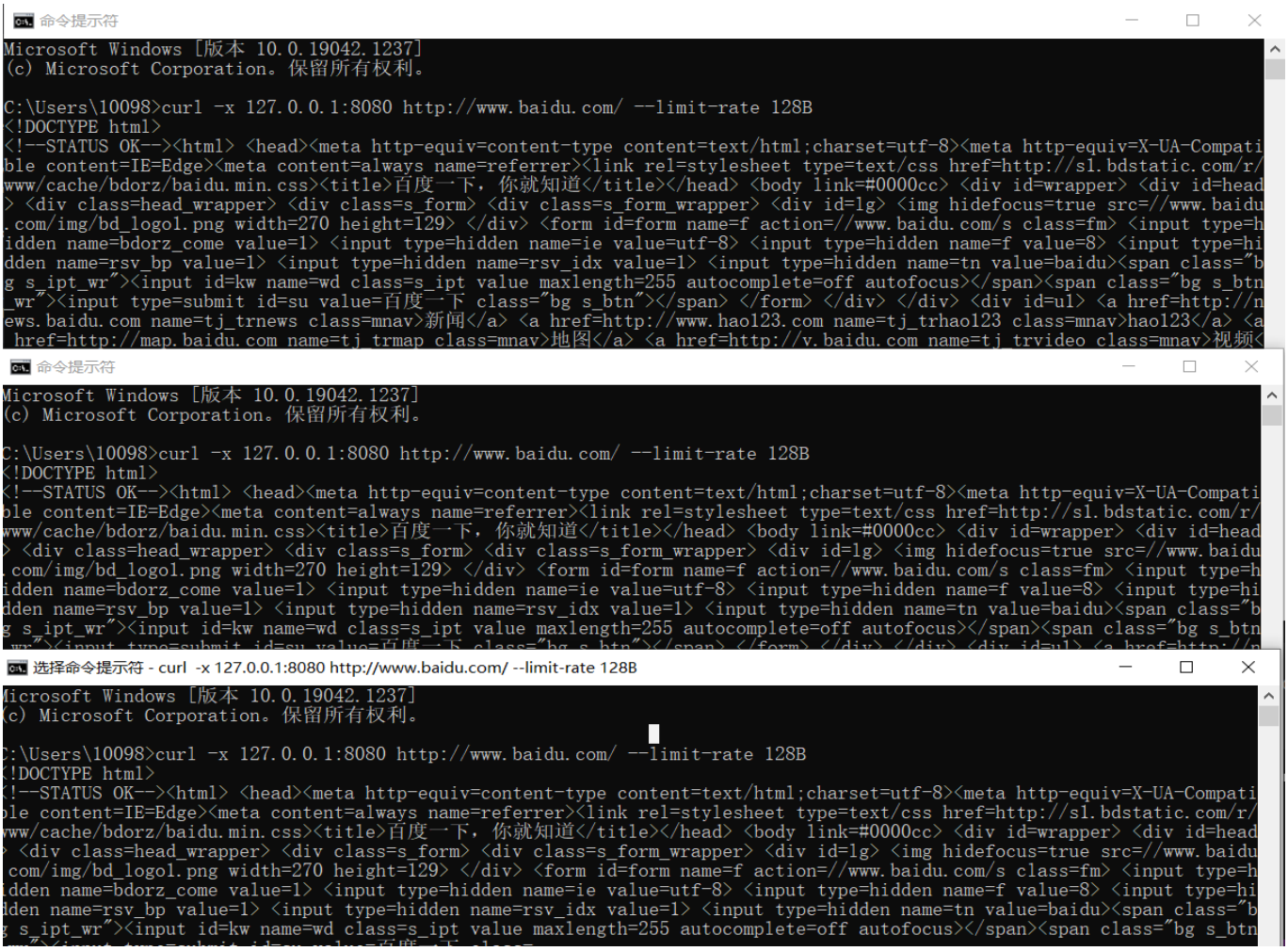
```
C:\Users\10098>curl -x 127.0.0.1:8080 http://www.baidu.com/ --head
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: private, no-cache, no-store, proxy-revalidate, no-transform
Connection: keep-alive
Content-Length: 277
Content-Type: text/html
Date: Sat, 09 Oct 2021 07:26:26 GMT
Etag: "575e1f65-115"
Last-Modified: Mon, 13 Jun 2016 02:50:13 GMT
Pragma: no-cache
Server: bfe/1.0.8.18
```

Proxy POST Request

```
C:\Users\10098>curl -x 127.0.0.1:8080 http://ecosimulation.com/cgi-bin/testpost.cgi -X POST -d "firstname=san&lastname=zhang"
hello!
key:lastname, value:zhang
key:firstname, value:san
goodbye!
```

Support Multithreading

To better display the result, the limit rate is changed to 128B



Caching

1. First request

The pycharm terminal display:

```
Ready to serve...
Received a connection from: ('127.0.0.1', 4137)
File does not exist in the cache
Ready to serve...
```

2. Following multiple requests

The pycharm terminal display: the web page seems to be modified multiple times during the test, so the modification status has switched repeatedly.

```
Ready to serve...
Received a connection from: ('127.0.0.1', 4168)
File exists in the cache
No need to modify
Read from cache
Ready to serve...
Received a connection from: ('127.0.0.1', 4173)
File exists in the cache
Need to modify
Read from cache
Ready to serve...
```

New file created in the same directory as the file *web_proxy.py* (on Desktop)