Networks Programming Assignment 1

MEMBERS:

1. Ibrahim Mohamed Elsayed Moustafa Omar | 21010023
2. Mohamed Mohmaed Mohamed Abdelmonaem | 21011213

DATASTRUCTURES:

No special data structures used.

OVERALL ARCHITECTURE:

Chunk consists of: chunkSize \r\n chunkData \r\n

RequestLine consists of: GET {filePath} HTTP/1.1 or

POST {filePath} HTTP/1.1

StatusLine consists of: HTTP/1.1 200 OK or

HTTP/1.1 404 Not Found

client_post

1 How to Run

```
py client_post.py {file_path} {server_ip} {server_port(optional)}
```

2 Functions

get_file_size(file) → returns the size of the file passed.

handle_post_text_file_request(url) → loads and parses the text file with path `url` and returns the list of responses that needs to be sent to the server either if the response is not chunked (i.e. len(responses list) = 1) or it is chunked (i.e. len(responses list) = 1 [requestLine and headers] + n [number of chunks] + 1 ['0\r\n\r\n' for the end of the file] = 2 + n).

handle_post_image_request(url) → loads and parses the image file with path `url` and returns the list of responses that needs to be sent to the server either if the response is not chunked (i.e. len(responses list) = 1) or it is chunked (i.e. len(responses list) = 1 [requestLine and headers] + n [number of chunks] + 1 ['0\r\n\r\n' for the end of the file] = 2 + n).

handle_post_request(url) → calls `handle_post_text_file_request` or `handle_post_image_request` according to the url extension.

3 Logic

Prepare the POST file and stores the requests which need to be sent to the server in `request` list, if `request` is None that means that the file path given is not found.

Loop on `request` list and send each item in it to the server.

A status is received from the server.

CLOSE request is sent to the server then CLOSED acknowledge is sent back to the client, what make the TCP connection between the client and the server close.

Client_get

1 How to run:

```
py client_get.py {file_path} {server_ip} {server_port(optional)}
```

2 Logic

A request is sent to the server with the file needed to be sent back from the server.

If no such file exists in the Server directory `FileNotFound404` response is sent.

If file is found it is sent to the client either in the single-response body or in several chunks after the status line and headers are sent in separate response.

Data sent is stored in a file with the same name of the main file in the server directory stored in the client directory.

Server [default port=80]

1 How to run:

```
py Server.py {server_port(optional)}
```

2 Functions

get_file_size(file) → returns the size of the file passed.

`parse_http_request(request)` → parses the request received into method, file path, http version, headers and body (in case of not-chunked).

`Handle_get_text_file_request(url)` → loads and parses the text file with path `url` and returns the list of responses that needs to be sent to the client either if the response is not chunked (i.e. len(responses list) = 1) or it is chunked (i.e. len(responses list) = 1 [requestLine and headers] + n [number of chunks] + 1 ['0\r\n\r\n' for the end of the file] = 2 + n).

`handle_get_image_request(url)` → loads and parses the image file with path `url` and returns the list of responses that needs to be sent to the client either if the response is not chunked (i.e. len(responses list) = 1) or it is chunked (i.e. len(responses list) = 1 [requestLine and headers] + n [number of chunks] + 1 ['0\r\n\r\n' for the end of the file] = 2 + n).

`handle_get_request(url)` → calls `handle_get_text_file_request` or `handle_get_image_request` according to the url extension.

`handle_post_request_not_chunked(filePath, headers, body)` → stores the body in the file of path `filePath`.

`handle_post_request_chunked(filePath, headers, channel)` → receives chunks to be saved in the file whose path is `filePath` from the client and stores them in the file.

`startWork(channel, address)` → the main function that runs for each thread (connection listened by the server).

3 Logic

For each connection listened by the server a thread is created for it `startWork` is called.

Receive the request from that client with the new connection.

In case of POST req:

If (`Transfer-Encoding` header is `chunked`):
    `handle_post_request_chunked` is called.
else:
    `handle_post_request_not_chunked` is called.

In case of GET req:

`handle_get_request` is called.

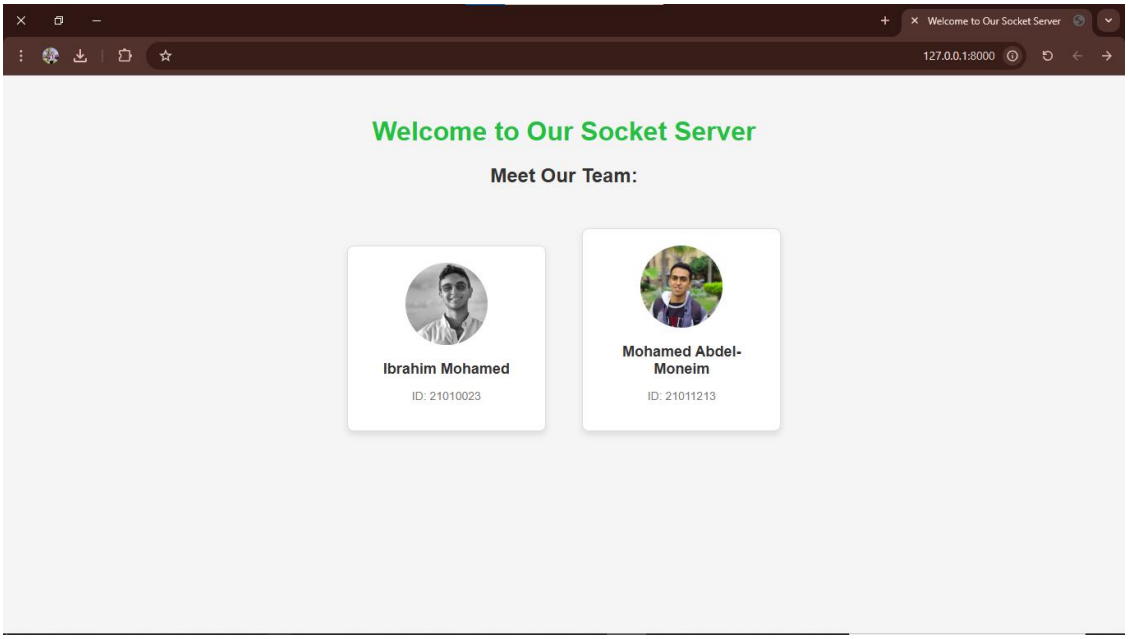
In case of CLOSE req:

The TCP connection is closed.

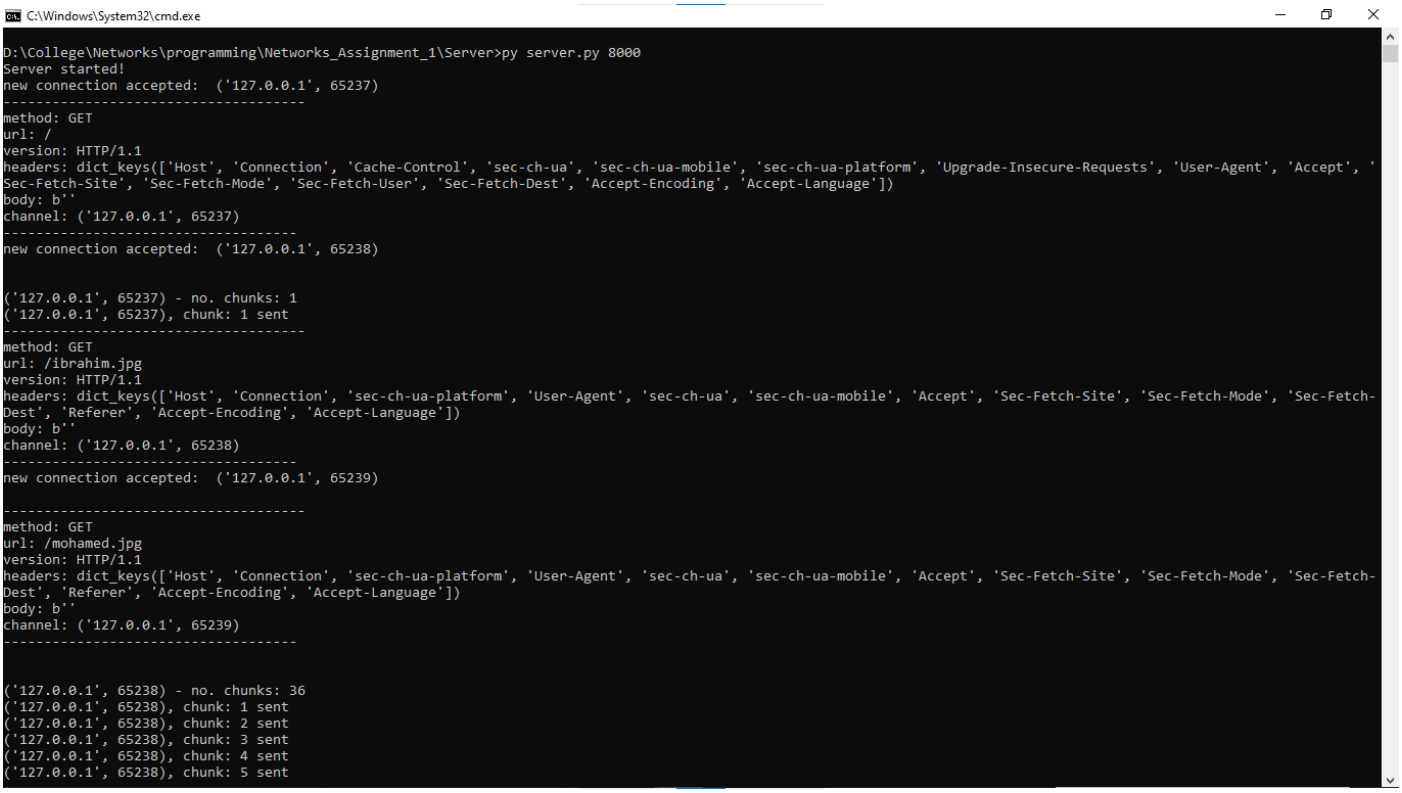
If the server is Idle (no connections (threads) exists) for `timeOut` time server is closed automatically.

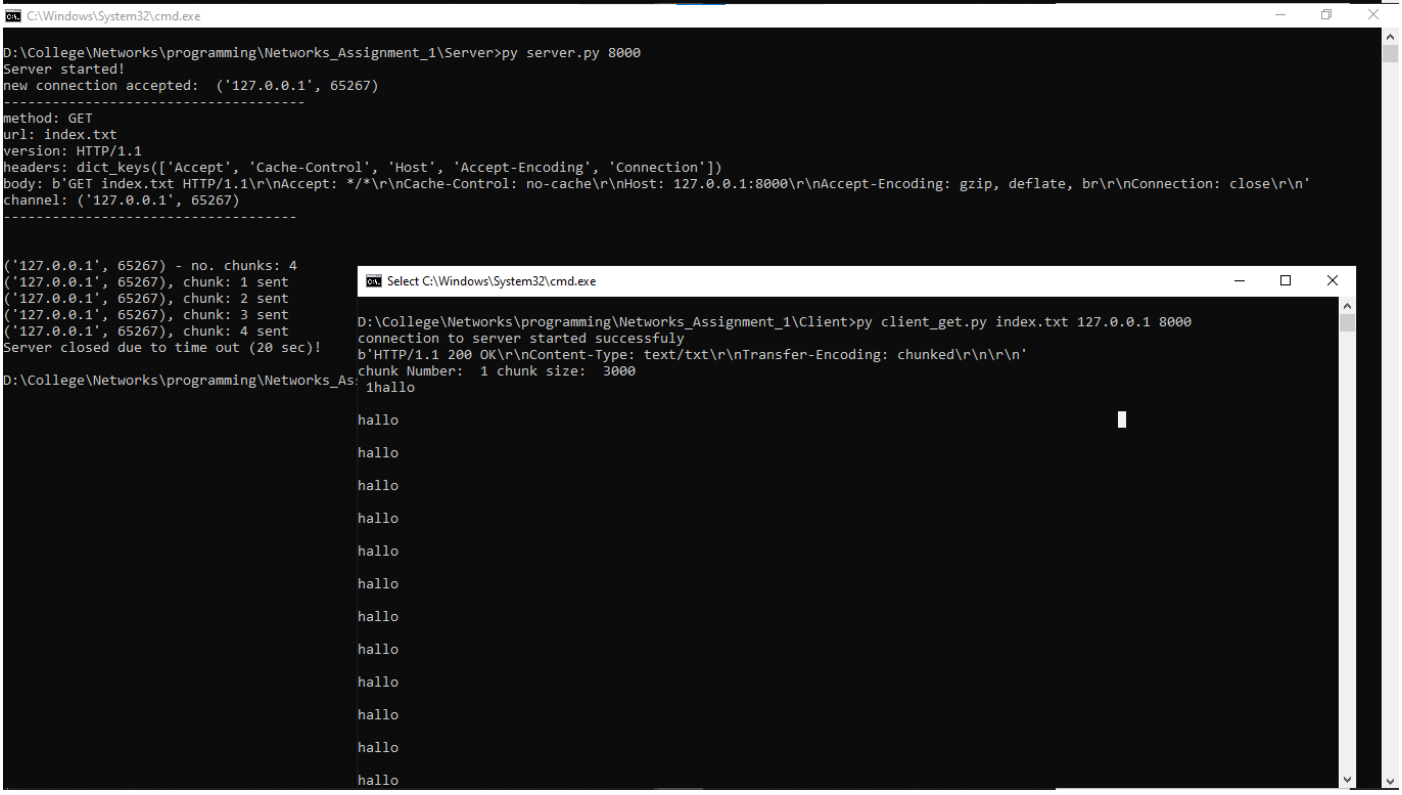
Git repository: https://github.com/zoaa3054/Networks_Assignment_1.git

BROWSER BONUS SCREENSHOTS:



TEST CASES:


C:\Windows\System32\cmd.exe
D:\College\Networks\programming\Networks_Assignment_1\Server>py server.py 8000
Server started!
new connection accepted: ('127.0.0.1', 65237)
method: GET
url: /
version: HTTP/1.1
headers: dict_keys(['Host', 'Connection', 'Cache-Control', 'sec-ch-ua', 'sec-ch-ua-mobile', 'sec-ch-ua-platform', 'Upgrade-Insecure-Requests', 'User-Agent', 'Accept', 'Sec-Fetch-Site', 'Sec-Fetch-Mode', 'Sec-Fetch-User', 'Sec-Fetch-Dest', 'Accept-Encoding', 'Accept-Language'])
body: b''
channel: ('127.0.0.1', 65237)
new connection accepted: ('127.0.0.1', 65238)
('127.0.0.1', 65237) - no. chunks: 1
('127.0.0.1', 65237), chunk: 1 sent
method: GET
url: /ibrahim.jpg
version: HTTP/1.1
headers: dict_keys(['Host', 'Connection', 'sec-ch-ua-platform', 'User-Agent', 'sec-ch-ua', 'sec-ch-ua-mobile', 'Accept', 'Sec-Fetch-Site', 'Sec-Fetch-Mode', 'Sec-Fetch-Dest', 'Referer', 'Accept-Encoding', 'Accept-Language'])
body: b''
channel: ('127.0.0.1', 65238)
new connection accepted: ('127.0.0.1', 65239)
method: GET
url: /mohamed.jpg
version: HTTP/1.1
headers: dict_keys(['Host', 'Connection', 'sec-ch-ua-platform', 'User-Agent', 'sec-ch-ua', 'sec-ch-ua-mobile', 'Accept', 'Sec-Fetch-Site', 'Sec-Fetch-Mode', 'Sec-Fetch-Dest', 'Referer', 'Accept-Encoding', 'Accept-Language'])
body: b''
channel: ('127.0.0.1', 65239)
('127.0.0.1', 65238) - no. chunks: 36
('127.0.0.1', 65238), chunk: 1 sent
('127.0.0.1', 65238), chunk: 2 sent
('127.0.0.1', 65238), chunk: 3 sent
('127.0.0.1', 65238), chunk: 4 sent
('127.0.0.1', 65238), chunk: 5 sent

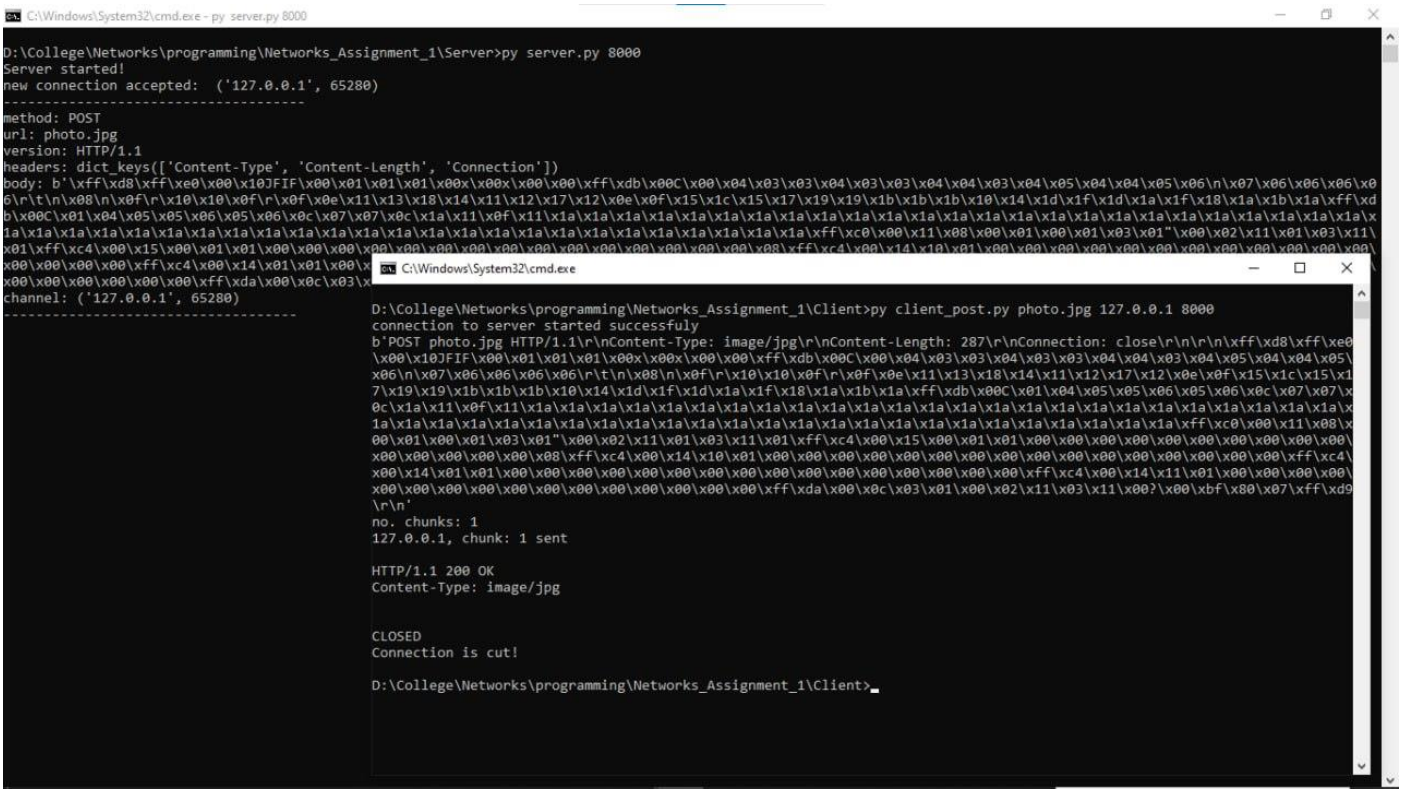

C:\Windows\System32\cmd.exe
D:\College\Networks\programming\Networks_Assignment_1\Server>py server.py 8000
Server started!
new connection accepted: ('127.0.0.1', 65267)
method: GET
url: index.txt
version: HTTP/1.1
headers: dict_keys(['Accept', 'Cache-Control', 'Host', 'Accept-Encoding', 'Connection'])
body: b'GET index.txt HTTP/1.1\r\nAccept: */*\r\nCache-Control: no-cache\r\nHost: 127.0.0.1:8000\r\nAccept-Encoding: gzip, deflate, br\r\nConnection: close\r\n'
channel: ('127.0.0.1', 65267)
('127.0.0.1', 65267) - no. chunks: 4
('127.0.0.1', 65267), chunk: 1 sent
('127.0.0.1', 65267), chunk: 2 sent
('127.0.0.1', 65267), chunk: 3 sent
('127.0.0.1', 65267), chunk: 4 sent
Server closed due to time out (20 sec)!
D:\College\Networks\programming\Networks_As
Select C:\Windows\System32\cmd.exe
D:\College\Networks\programming\Networks_Assignment_1\Client>py client_get.py index.txt 127.0.0.1 8000
connection to server started successfuly
b'HTTP/1.1 200 OK\r\nContent-Type: text/txt\r\nTransfer-Encoding: chunked\r\n\r\n'
chunk Number: 1 chunk size: 3000
1hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo
hallo

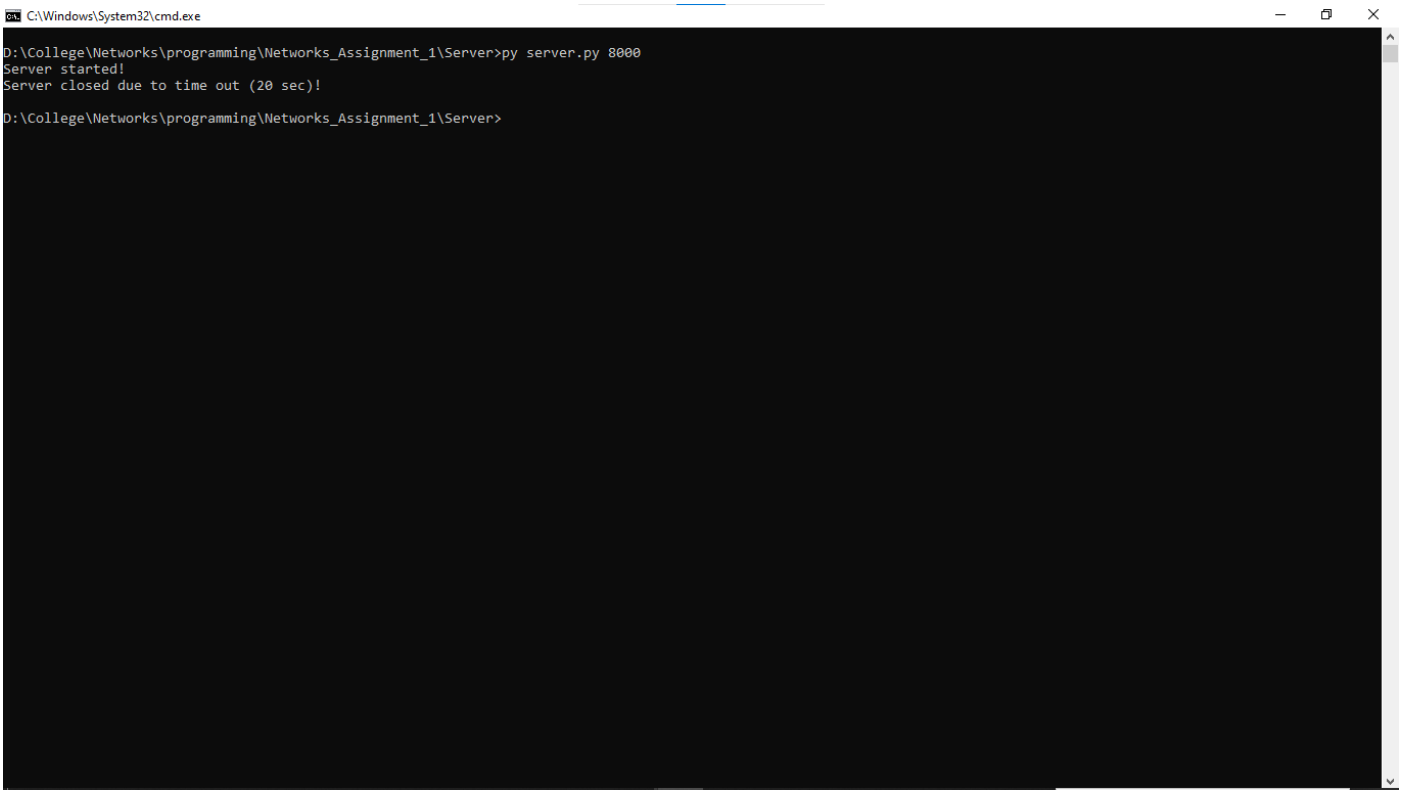
C:\Windows\System32\cmd.exe

```
D:\College\Networks\programming\Networks_Assignment_1\Server>py server.py 8000
Server started!
new connection accepted:  ('127.0.0.1', 65280)
-------------------------------------
method: POST
url: photo.jpg
version: HTTP/1.1
headers: dict_keys(['Content-Type', 'Content-Length', 'Connection'])
body: b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00x\x00x\x00\x00\xff\xdb\x00C\x00\x04\x03\x03\x04\x03\x03\x04\x04\x03\x04\x05\x04\x04\x05\x06\n\x07\x06\x06\x06\x0
6\r\t\n\x08\n\x0f\r\x10\x10\x0f\r\x0f\x0e\x11\x13\x18\x14\x11\x12\x17\x12\x0e\x0f\x15\x1c\x15\x17\x19\x19\x1b\x1b\x1b\x10\x14\x1d\x1f\x1d\x1a\x1f\x18\x1a\x1b\x1a\xff\xd
b\x00C\x01\x04\x05\x05\x06\x05\x06\x0c\x07\x07\x0c\x1a\x11\x0f\x11\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x
1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\xff\xc0\x00\x11\x08\x00\x01\x00\x01\x03\x01"\x00\x02\x11\x01\x03\x11\
x01\xff\xc4\x00\x15\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\xff\xc4\x00\x14\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xc4\x00\x14\x11\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\xff\xda\x00\x0c\x03\x01\x00\x02\x11\x03\x11\x00?\x00\xbf\x80\x07\xff\xd9\r\n'
channel: ('127.0.0.1', 65280)
-------------------------------------

Server closed due to time out (20 sec)!

D:\College\Networks\programming\Networks_Assignment_1\Server>
```

C:\Windows\System32\cmd.exe

```
('127.0.0.1', 65239), chunk: 8 sent
('127.0.0.1', 65239), chunk: 9 sent
('127.0.0.1', 65239), chunk: 10 sent
('127.0.0.1', 65239), chunk: 11 sent
('127.0.0.1', 65239), chunk: 12 sent
('127.0.0.1', 65239), chunk: 13 sent
('127.0.0.1', 65239), chunk: 14 sent
('127.0.0.1', 65239), chunk: 15 sent
('127.0.0.1', 65239), chunk: 16 sent
('127.0.0.1', 65239), chunk: 17 sent
('127.0.0.1', 65239), chunk: 18 sent
('127.0.0.1', 65239), chunk: 19 sent
('127.0.0.1', 65239), chunk: 20 sent
('127.0.0.1', 65239), chunk: 21 sent
('127.0.0.1', 65239), chunk: 22 sent
('127.0.0.1', 65239), chunk: 23 sent
('127.0.0.1', 65239), chunk: 24 sent
('127.0.0.1', 65239), chunk: 25 sent
('127.0.0.1', 65239), chunk: 26 sent
('127.0.0.1', 65239), chunk: 27 sent
('127.0.0.1', 65239), chunk: 28 sent
('127.0.0.1', 65239), chunk: 29 sent
('127.0.0.1', 65239), chunk: 30 sent
('127.0.0.1', 65239), chunk: 31 sent
('127.0.0.1', 65239), chunk: 32 sent
('127.0.0.1', 65239), chunk: 33 sent
('127.0.0.1', 65239), chunk: 34 sent
('127.0.0.1', 65239), chunk: 35 sent
('127.0.0.1', 65239), chunk: 36 sent
('127.0.0.1', 65239), chunk: 37 sent
('127.0.0.1', 65239), chunk: 38 sent
('127.0.0.1', 65239), chunk: 39 sent
('127.0.0.1', 65239), chunk: 40 sent
('127.0.0.1', 65239), chunk: 41 sent
('127.0.0.1', 65239), chunk: 42 sent
('127.0.0.1', 65239), chunk: 43 sent
('127.0.0.1', 65239), chunk: 44 sent
('127.0.0.1', 65237) channel terminated due to timeout
threads:  3
Server closed due to time out (20 sec)!
('127.0.0.1', 65238) channel terminated due to timeout
threads:  2
('127.0.0.1', 65239) channel terminated due to timeout
threads:  1

D:\College\Networks\programming\Networks_Assignment_1\Server>
```


C:\Windows\System32\cmd.exe - py server.py 8000
D:\College\Networks\programming\Networks_Assignment_1\Server>py server.py 8000
Server started!
new connection accepted: ('127.0.0.1', 65280)
method: POST
url: photo.jpg
version: HTTP/1.1
headers: dict_keys(['Content-Type', 'Content-Length', 'Connection'])
channel: ('127.0.0.1', 65280)
C:\Windows\System32\cmd.exe
D:\College\Networks\programming\Networks_Assignment_1\Client>py client_post.py photo.jpg 127.0.0.1 8000
connection to server started successfuly
no. chunks: 1
127.0.0.1, chunk: 1 sent
HTTP/1.1 200 OK
Content-Type: image/jpg
CLOSED
Connection is cut!
D:\College\Networks\programming\Networks_Assignment_1\Client>


C:\Windows\System32\cmd.exe
D:\College\Networks\programming\Networks_Assignment_1\Server>py server.py 8000
Server started!
Server closed due to time out (20 sec)!
D:\College\Networks\programming\Networks_Assignment_1\Server>