

Computer Networks

CSE351

Nachaat fahmy Nachaat	19p2460
------------------------------	----------------

Shady Osama Wadeea	19P2602
---------------------------	----------------

Kirollos Georges Boutros	19p9058
---------------------------------	----------------

THE CODE

```
import socket
import sys

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.socket(socket.AF_INET, socket.SOCK_STREAM)

# Fill in start.
recv_buffer = 4096
TCP_IP = "localhost"
TCP_PORT = 8888
tcpSerSock.bind((TCP_IP, TCP_PORT))
tcpSerSock.listen(2)
print("Listening on port: ", TCP_PORT)

# Fill in end.

while 1:
    # Start receiving data from the client
    print ('\n\nReady to serve...')
    tcpCliSock, addr = tcpSerSock.accept() # return address and tcp client
socket
    print ('Received a connection from:', addr)
    # fill in start
    message = tcpCliSock.recv(4096)
    # fill in end
    if message == "":
        continue
    print (message)
    # Extract the filename from the given message
    file = message.split()[1]
    filename = file.split('/')[1]
    fileExist = "false"
    filetouse = file
    #open file url
    f1 = open("urlblock.txt")
```

```

flag = 1
for x in f1:

    if x == filename:

        flag = -1
        header = 'HTTP/1.0 404 Not Found\n\n'
        response = '<html>' \
                    '<h1>' \
                    'Error' \
                    '403: File' \
                    'blocked' \
                    '</h1>' \
                    '</html>'
        tcpCliSock.send(header.encode() + response.encode())

if flag == -1:
    continue

f1.close()
try:
    # Check whether the file exist in the cache
    f = open(filetouse[1:], "rb")
    outputdata = f.read()
    fileExist = "true"

    # ProxyServer finds a cache hit and generates a response message
    tcpCliSock.sendall("HTTP/1.0 200 OK\r\n".encode())
    tcpCliSock.sendall("Content-Type:text/html\r\n".encode())
    tcpCliSock.sendall("Content-Type: image/jpeg\r\n".encode())
    # Fill in start.
    tcpCliSock.sendall(outputdata)
    # Fill in end.
    f.close()
    print('Read from cache')

# Error handling for file not found in cache
except IOError:
    if fileExist == "false":
        print("hi")
        file = file[1:]
        #hostn = file
        # Create a socket on the proxyserver

```

```

c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
hostname = file.replace("www.", "", 1)
print(hostname +
"=====")
try:
    # Connect to the socket to port 80
    # Fill in start.
    fileobj = c.makefile('rwb', 0)

    print("////////////////////////////////////////:")
    if not ("Referer" in message):
        print("*Connecting to server"+ hostname)
        # Connect to the socket to port 80
        c.connect((hostname, 80))
        print(hostname)
        con = hostname
        fileobj.write(b'GET / HTTP/1.0\r\n\r\n')
    else:
        print("**Get path in referer: " + hostname)
        c.connect((con, 80))
        fileobj.write(b'GET /' + hostname + '
HTTP/1.0\r\n\r\n'.encode())
    # Read the response into buffer
    # Fill in start.
    print("done")
    buff = fileobj.read()

    print("done2")
    # 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")
    print(buff)
    for i in range(0, len(buff)):
        tmpFile.write(buff[i])

    print("done3")
    tcpCliSock.sendall("HTTP/1.0 200 OK\r\n".encode())
    tcpCliSock.sendall("Content-Type:text/html\r\n".encode())
    tcpCliSock.sendall("Content-Type: image/jpeg\r\n".encode())
    print("done4")

```

```

        tcpCliSock.sendall(buff)
        #print(buff)
        # Fill in end.
        print('sent to client')
        tmpFile.close()
    except socket.gaierror:
        header = 'HTTP/1.0 404 Not Found\n\n'
        response = '<html>' \
            '<h1>' \
            '    Error    ' \
            '    404: File' \
            '    not found ' \
            '</h1>' \
            '</html>'
        tcpCliSock.send(header.encode() + response.encode())

    except IOError as e:
        print("Illegal request")
        print(e.args)

    # except socket.gaierror:
    #     open(error.txt)
    #     tcpCliSock.sendall(file.read)

else:
    # HTTP response message for file not found
    # Fill in start.
    header = 'HTTP/1.0 404 Not Found\n\n'
    response = '<html>' \
        '<h1>' \
        '    Error    ' \
        '    404: File' \
        '    not found ' \
        '</h1>' \
        '</html>'
    tcpCliSock.send(header.encode() + response.encode())
    # Fill in end.
    # Close the client and the server sockets

tcpCliSock.close()

# Fill in start.
tcpSerSock.close()

```


This is a Python script that creates a simple HTTP proxy server. It listens for incoming connections from clients, and when it receives a connection, it makes a request to the specified server and returns the response to the client.

Here is an overview of the script:

```
from socket import *
import sys

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)
tcpSerSock.bind((sys.argv[1], 8888))
tcpSerSock.listen(100)

while 1:
    # Start receiving data from the client
    print('Ready to serve...')
    tcpCliSock, addr = tcpSerSock.accept()
    print('Received a connection from:', addr)
    message = tcpCliSock.recv(1024).decode("utf-8")
    print(message)
```

- It creates a server socket, binds it to the specified IP address and port (8888), and starts listening for incoming connections.
- It enters a loop where it waits for a client connection using the `accept()` method of the server socket. When a connection is received, it stores the client socket and client address in variables `tcpCliSock` and `addr`, respectively

- It receives data from the client socket using the `recv()` method and stores it in the `message` variable. It then extracts the filename from the message and stores it in the `filename` variable.

```
print (message)
# Extract the filename from the given message
file = message.split()[1]
filename = file.split('/')[1]
fileExist = "false"
filetouse = file

try:
    # Check whether the file exist in the cache
    f = open(filetouse[1:], "rb")
    outputdata = f.read()
    fileExist = "true"
```

Terminal output

```
PS D:\semester 7> cd networks
PS D:\semester 7\networks> python networks.py 127.0.0.1
Ready to serve...
Received a connection from: ('127.0.0.1', 62362)
GET /google.com HTTP/1.1
Host: 127.0.0.1:8888
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

- It checks whether the file is present in the cache by trying to open it in read mode. If the file is present, it reads the contents of the file into the output data variable and sends an HTTP response message along with the file contents to the client.

```
try:
    # Check whether the file exist in the cache
    f = open(filetouse[1:], "rb")
    outputdata = f.read()
    fileExist = "true"

    # ProxyServer finds a cache hit and generates a response message
    tcpCliSock.sendall("HTTP/1.0 200 OK\r\n".encode())
    tcpCliSock.sendall("Content-Type:text/html\r\n".encode())
    tcpCliSock.sendall("Content-Type: image/jpeg\r\n".encode())
    # Fill in start.
    tcpCliSock.sendall(outputdata)
    # Fill in end.
    f.close()
    print('Read from cache')

# Error handling for file not found in cache
```

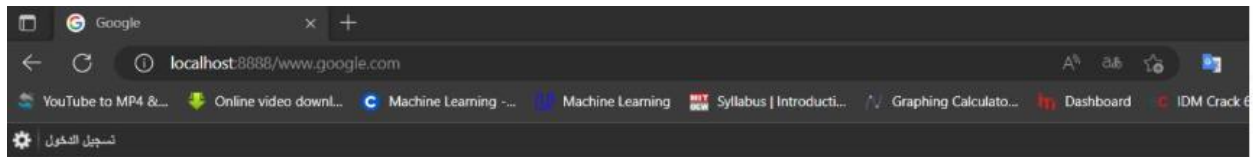
Terminal Output

```
C:\> Command Prompt

Cache-Control: max-age=0
sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108", "Google Chrome";v="108"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

Read from cache
```

This means that we are reading from the cache the input of the web page



بحث متقدم



ضربة حظ

بحث Google

محرّك بحث Google متوفّر باللغة: English

حلول الشركات كل ما تحب محرّكه عن Google هنا Google.com.eg


© 2023 - الخصوصية - البريد

- If we try to reach google for the first time from using our proxy server
- If the file is not present in the cache, it creates a new socket and connects to the server specified in the filename. It then sends an HTTP request for the file to the server and stores the response in the buff variable. It also creates a new file in the cache and writes the contents of the buff variable to the file.

```
except IOError:
    if fileExist == "false":
        print("hi")
        file = file[1:]
        #hostn = file
        # Create a socket on the proxyserver
        print ("creating socket on proxy server")
        c = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        hostname = file.replace("www.", "", 1)
        print(hostname + "=====")
        try:
            # Connect to the socket to port 80
            print ("connected to the socket to port 80")
            # Fill in start.
            fileobj = c.makefile('rwb', 0)

            print("////////////////////////////////////:")
            if not ("Referer" in message):
                print("*Connecting to server"+ hostname)
                # Connect to the socket to port 80
                c.connect((hostname, 80))
                print(hostname)
                con = hostname
                fileobj.write(b'GET / HTTP/1.0\r\n\r\n')
```

Cached copy of requested file:

 www.google.com

1/8/2023 11:26 AM

MS-DOS Applicati...

52 KB

Terminal output

```
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

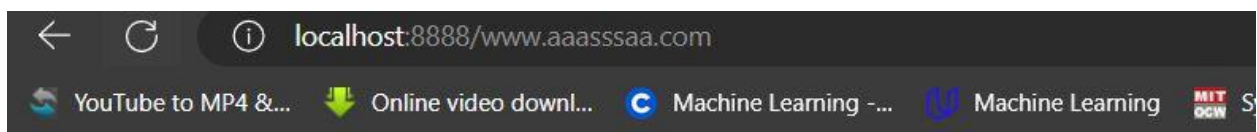
hi
creating socket on proxy server
google.com=====
connected to the socket to port 80
/////////////////////////////////////:
*Connecting to servergoogle.com
google.com
done
done2
HTTP/1.0 200 OK
Date: Sun, 08 Jan 2023 19:26:26 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Cross-Origin-Opener-Policy-Report-Only: same-origin-allow-popups; report-to="gws"
Report-To: {"group":"gws","max_age":2592000,"endpoints":[{"url":"https://csp.with
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
```

Error Handling

If there is an error while trying to open the file in the cache or connect to the server, it sends an HTTP error message to the client.

```
else:
    # HTTP response message for file not found
    # Fill in start.
    header = 'HTTP/1.0 404 Not Found\n\n'
    response = '<html>' \
        '<h1>' \
        'Error' \
        '404: File' \
        'not found' \
        '</h1>' \
        '</html>'
    tcpCliSock.send(header.encode() + response.encode())
    # Fill in end.
    # Close the client and the server sockets

    tcpCliSock.close()
```



Error 404: File not found

Block sites or filtering sites from the proxy server

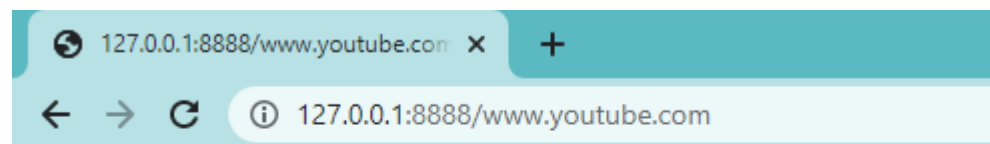
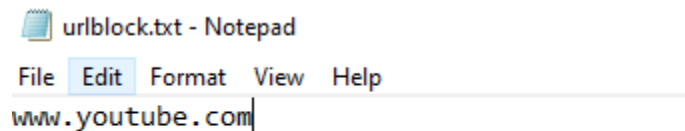
We have a text document that contains sites to filter

```
f1 = open("urlblock.txt")
flag = 1
for x in f1:

    if x == filename:

        flag = -1
        header = 'HTTP/1.0 404 Not Found\n\n'
        response = '<html>' \
                    '    <h1>' \
                    '        Error    ' \
                    '    403: File' \
                    '        blocked    ' \
                    '</h1>' \
                    '</html>'
        tcpCliSock.send(header.encode() + response.encode())
```

So we have the text document



Error 403: File blocked