PROGRES-2020: Mini-Project 1: Proxy TCP and Proxy HTTP

Framing: Sébastien TIXEUIL Sebastien.Tixeuil@lip6.fr
Student:
BOUCHAMA NASSIM

Project design:

Mini-Project 1: Proxy TCP and Proxy HTTP

Part 1: Proxy TCP

We have to create an architecture (Client-server) for a data transmission, and this is necessarily going through a Proxy. The latter will play the role of a server when it communicates with a client, and then it will be a client itself when it communicates with the server that we are trying to reach.

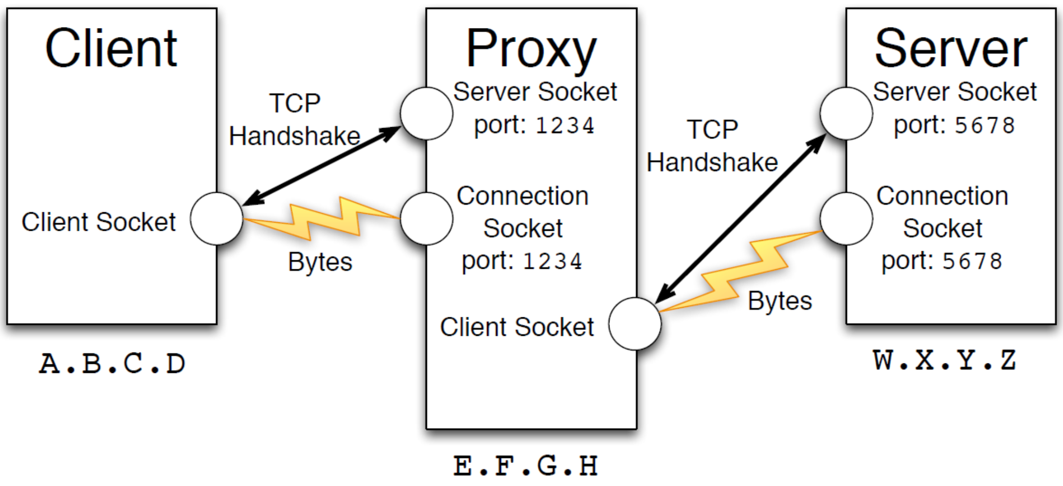


Figure 1: Proxy TCP Operating Principle

A TCP connection is established between the client and the proxy and then between the proxy and the server and we go through all the important steps of the TCP transport layer protocol, including the 3-way-handshake to establish the connection and open a lane before sending the data in gross, TCP is a protocol oriented connection with the concept of ports (multiplexing multiplying) ,a connection is established above all sends data between two entities this knowing the port number that one seeks to reach.

1. Analysis of the Proxy TCP code:

```
from socket import *
from threading import *
```

- We import the libraries we want to use, the socket library for all **functions** that relate to TCP communications sockets, and the **threading** library for future use that creates sub-processes in the server to manage customer queries.

```
Proxyport=1234
Serverport=5678
Servername='127.0.0.1'
```

- The proxy and server port numbers and its IP address (here local) are backed up within the proxy memory.

```
ProxySocket=socket(AF_INET,SOCK_STREAM)
```

- A TCP-type connection proxy sockette (Stream) is created

```
ProxySocket.bind(('',Proxyport))
ProxySocket.listen(5)
print('proxy ready')
```

- The proxy sockette is asked to wait to receive a connection request on its port (Proxy port), this for any IP address ('')

```
def handle_client(clientSocket):
```

- The function of the proxy by being a client for the server is defined here.

```
while True :
received=clientSocket.recv(4096)
```

- The received variable records everything the customer sockett receives from the connection socket set below in the code

```
if not received :
clientSocket.close()
```

- and if it receives nothing, we might as well close this sockett to say that there is no communication.

```
else :
tempSocket=socket (AF_INET,SOCK_STREAM)
```

- In the event that the customer sockett receives a data, a local socket is created called a temporary socket and gives it the functionality of a full-fledged client.

```
tempSocket.connect((Servername,Serverport))
tempSocket.send(received)
message=tempSocket.recv(4096)
clientSocket.sendall(message)
tempSocket.close()
```

- The socket connects with the server sockette (handshake)
- Sends her what she has received previously from the connection sockette
- Waits to receive the message from the server after it has made a change to the original message (the server just capitalizes the message received)
- And in the end, this temporary socket that plays the role of client sends everything it received to the connection sockette
- And we close this sockett until the next communication

```
while True :
        connectionSocket, address=ProxySocket.accept()
        handle_client(connectionSocket)
```

- This is where the connection sockette is created that receives the request to be processed from the customer, which sends it to the temporary proxy sockett that makes the function handle_client explained above and which finally gives a response to the customer.

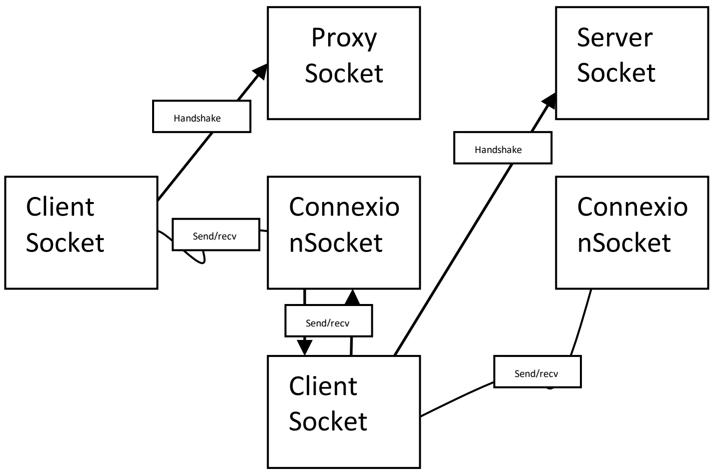


Figure 2: Summary of steps performed by TCP proxy sockets

2. Program test:

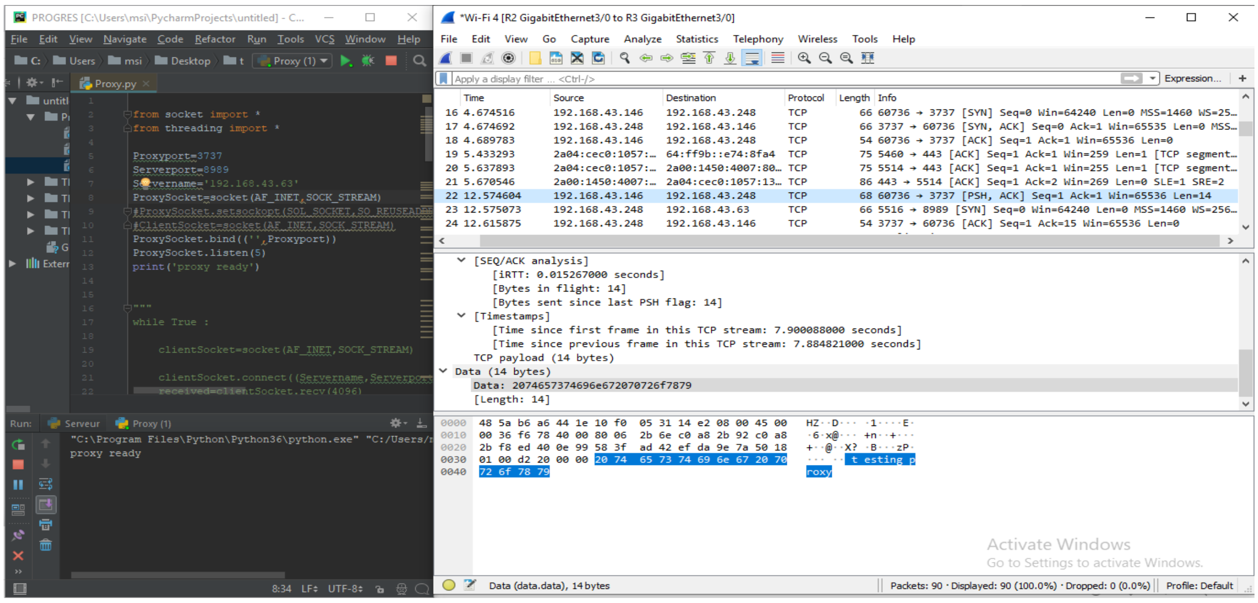
We test the program either by creating a client and a server on python or by using a web server like **apache** and a web client like **firefox**.

Mini-Project 1: Proxy TCP and Proxy HTTP

In this case, we use a client and a server created beforehand in a TP for the test and we analyze a trace of the frame sent with wireshark (we create a small LAN with 3 computers of my comrades who play the role of client, proxy and server)

Trace analysis with wireshark:

- By putting a probe on the wifi map of server and proxy machines, we can see the details of the frame transmitted via the TCP protocol as well as the message (testing proxy) received in request from the client and that goes through the proxy and then reaches the server that processes it and then returns it to the customer through the proxy.



PROGRES [C:\Users\msi\PycharmProjects\untitled] - C...

Proxy.py

```python
from socket import *
from threading import *

Proxyport=3737
Serverport=8989
Servername='192.168.43.63'
ProxySocket=socket(AF_INET,SOCK_STREAM)
#ProxySocket.setsockopt(SOL_SOCKET,SO_REUSEADDR
#ClientSocket=socket(AF_INET,SOCK_STREAM)
ProxySocket.bind(('',Proxyport))
ProxySocket.listen(5)
print('proxy ready')


"""
while True :

    clientSocket=socket(AF_INET,SOCK_STREAM)

    clientSocket.connect((Servername,Serverport
    received=clientSocket.recv(4096)
```

```
"C:\Program Files\Python\Python36\python.exe" "C:/Users/
proxy ready
```

*Wi-Fi 4 [R2 GigabitEthernet3/0 to R3 GigabitEthernet3/0]

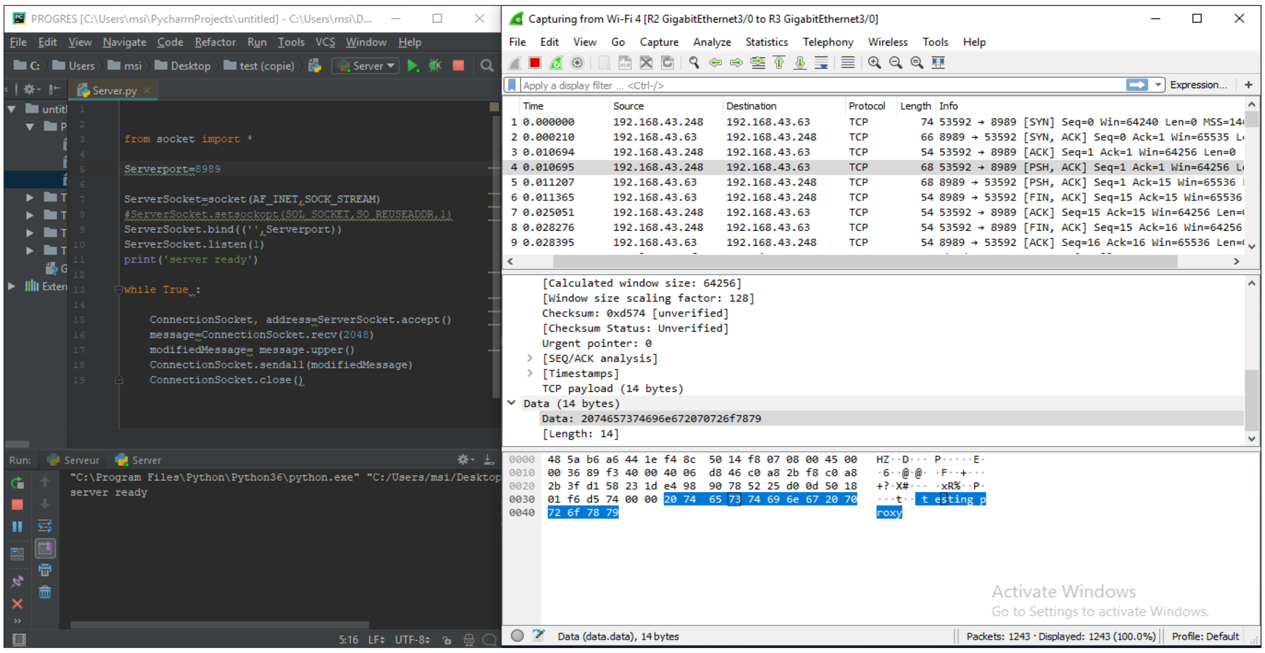
```
    Time         Source              Destination         Protocol  Length Info
16 4.674516     192.168.43.146      192.168.43.248      TCP       66 60736 → 3737 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=25…
17 4.674692     192.168.43.248      192.168.43.146      TCP       66 3737 → 60736 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS…
18 4.689783     192.168.43.146      192.168.43.248      TCP       54 60736 → 3737 [ACK] Seq=1 Ack=1 Win=65536 Len=0
19 5.433293     2a04:cec0:1057:…    64:ff9b::e74:8fa4   TCP       75 5460 → 443 [ACK] Seq=1 Ack=1 Win=259 Len=1 [TCP segment…
20 5.637893     2a04:cec0:1057:…    2a00:1450:4007:80…  TCP       75 5514 → 443 [ACK] Seq=1 Ack=1 Win=255 Len=1 [TCP segment…
21 5.670546     2a00:1450:4007:…    2a04:cec0:1057:13…  TCP       86 443 → 5514 [ACK] Seq=1 Ack=2 Win=269 Len=0 SLE=1 SRE=2
22 12.574604    192.168.43.146      192.168.43.248      TCP       68 60736 → 3737 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=14
23 12.575073    192.168.43.248      192.168.43.63       TCP       66 5516 → 8989 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256…
24 12.615875    192.168.43.248      192.168.43.146      TCP       54 3737 → 60736 [ACK] Seq=1 Ack=15 Win=65536 Len=0
```

```
  [SEQ/ACK analysis]
     [iRTT: 0.015267000 seconds]
     [Bytes in flight: 14]
     [Bytes sent since last PSH flag: 14]
  [Timestamps]
     [Time since first frame in this TCP stream: 7.900088000 seconds]
     [Time since previous frame in this TCP stream: 7.884821000 seconds]
  TCP payload (14 bytes)
Data (14 bytes)
   Data: 2074657374696e672070726f7879
   [Length: 14]
```

```
0000  48 5a b6 a6 44 1e 10 f0  05 31 14 e2 08 00 45 00   HZ··D··· ·1····E·
0010  00 36 f6 78 40 00 80 06  2b 6e c0 a8 2b 92 c0 a8   ·6·x@··· +n··+···
0020  2b f8 ed 40 0e 99 58 3f  ad 42 ef da 9e 7a 50 18   +··@··X? ·B···zP·
0030  01 00 d2 20 00 00 20 74  65 73 74 69 6e 67 20 70   ······ t esting p
0040  72 6f 78 79                                        roxy
```

Data (data.data), 14 bytes | Packets: 90 · Displayed: 90 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

Figure 3: trace analysis within the proxy



PROGRES [C:\Users\msi\PycharmProjects\untitled] - C:\Users\msi\D...

Server.py

```python


from socket import *

Serverport=8989

ServerSocket=socket(AF_INET,SOCK_STREAM)
#ServerSocket.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
ServerSocket.bind(('',Serverport))
ServerSocket.listen(1)
print('server ready')

while True :

    ConnectionSocket, address=ServerSocket.accept()
    message=ConnectionSocket.recv(2048)
    modifiedMessage= message.upper()
    ConnectionSocket.sendall(modifiedMessage)
    ConnectionSocket.close()
```

```
"C:\Program Files\Python\Python36\python.exe" "C:/Users/msi/Desktop
server ready
```

Capturing from Wi-Fi 4 [R2 GigabitEthernet3/0 to R3 GigabitEthernet3/0]

```
   Time         Source              Destination         Protocol  Length Info
1 0.000000     192.168.43.248      192.168.43.63       TCP       74 53592 → 8989 [SYN] Seq=0 Win=64240 Len=0 MSS=14
2 0.000210     192.168.43.63       192.168.43.248      TCP       66 8989 → 53592 [SYN, ACK] Seq=0 Ack=1 Win=65535 L
3 0.010694     192.168.43.248      192.168.43.63       TCP       54 53592 → 8989 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4 0.010695     192.168.43.248      192.168.43.63       TCP       68 53592 → 8989 [PSH, ACK] Seq=1 Ack=1 Win=64256 L
5 0.011207     192.168.43.63       192.168.43.248      TCP       68 8989 → 53592 [PSH, ACK] Seq=1 Ack=15 Win=65536
6 0.011365     192.168.43.63       192.168.43.248      TCP       54 8989 → 53592 [FIN, ACK] Seq=15 Ack=15 Win=65536
7 0.025051     192.168.43.248      192.168.43.63       TCP       54 53592 → 8989 [ACK] Seq=15 Ack=15 Win=64256 Len=
8 0.028276     192.168.43.248      192.168.43.63       TCP       54 53592 → 8989 [FIN, ACK] Seq=15 Ack=16 Win=64256
9 0.028395     192.168.43.63       192.168.43.248      TCP       54 8989 → 53592 [ACK] Seq=16 Ack=16 Win=65536 Len=
```

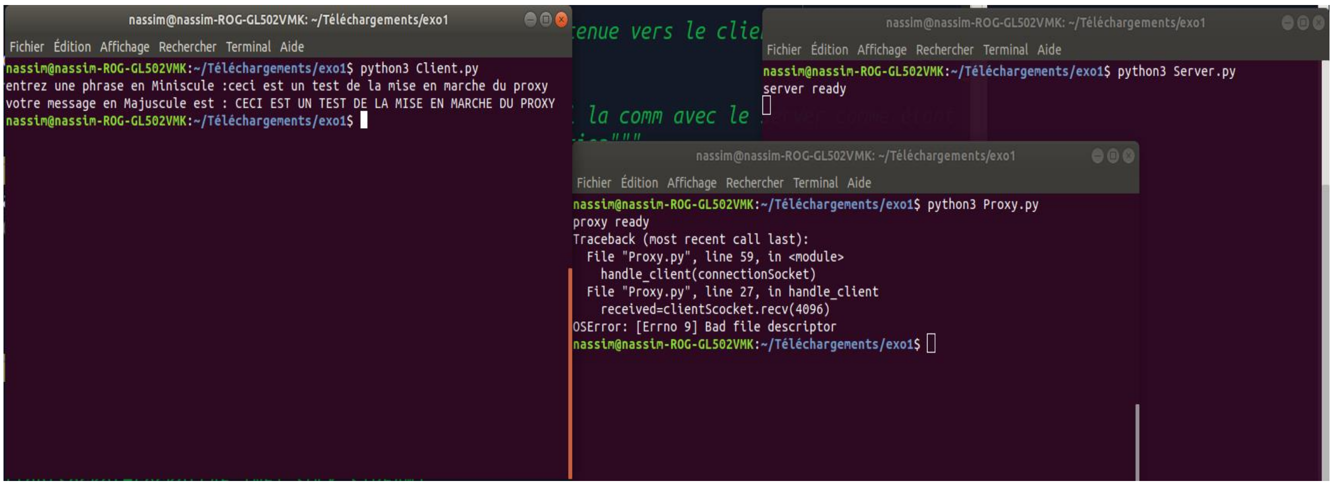
```
     [Calculated window size: 64256]
     [Window size scaling factor: 128]
     Checksum: 0xd574 [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   > [SEQ/ACK analysis]
   > [Timestamps]
     TCP payload (14 bytes)
Data (14 bytes)
     Data: 2074657374696e672070726f7879
     [Length: 14]
```

```
0000  48 5a b6 a6 44 1e f4 8c  50 14 f8 07 08 00 45 00   HZ··D··· P·····E·
0010  00 36 89 f3 40 00 40 06  d8 46 c0 a8 2b f8 c0 a8   ·6··@·@· ·F··+···
0020  2b 3f d1 58 23 1d e4 98  90 78 52 25 d0 0d 50 18   +?·X#··· ·xR%··P·
0030  01 f6 d5 74 00 00 20 74  65 73 74 69 6e 67 20 70   ···t·· t esting p
0040  72 6f 78 79                                        roxy
```

Data (data.data), 14 bytes | Packets: 1243 · Displayed: 1243 (100.0%) | Profile: Default

Figure 4: Track analysis within the server

Mini-Project 1: Proxy TCP and Proxy HTTP



```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Client.py
entrez une phrase en Miniscule :ceci est un test de la mise en marche du proxy
votre message en Majuscule est : CECI EST UN TEST DE LA MISE EN MARCHE DU PROXY
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

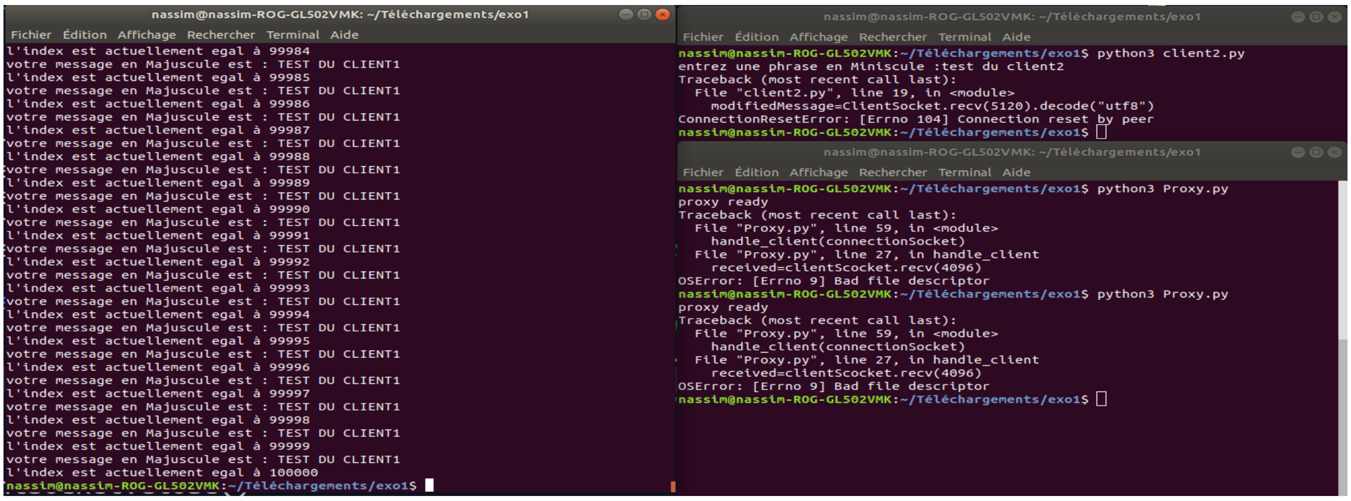
```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Server.py
server ready
```

```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Proxy.py
proxy ready
Traceback (most recent call last):
  File "Proxy.py", line 59, in <module>
    handle_client(connectionSocket)
  File "Proxy.py", line 27, in handle_client
    received=clientScocket.recv(4096)
OSError: [Errno 9] Bad file descriptor
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

<u>Figure</u> 5: Testing how to send a customer query

3. Testing the program with several clients:

In this test we launch a loop where the customer1 makes severalconsecutive requests and we try to



```
l'index est actuellement egal à 99984
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99985
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99986
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99987
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99988
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99989
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99990
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99991
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99992
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99993
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99994
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99995
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99996
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99997
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99998
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 99999
votre message en Majuscule est : TEST DU CLIENT1
l'index est actuellement egal à 100000
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 client2.py
entrez une phrase en Miniscule :test du client2
Traceback (most recent call last):
  File "client2.py", line 19, in <module>
    modifiedMessage=ClientSocket.recv(5120).decode("utf8")
ConnectionResetError: [Errno 104] Connection reset by peer
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Proxy.py
proxy ready
Traceback (most recent call last):
  File "Proxy.py", line 59, in <module>
    handle_client(connectionSocket)
  File "Proxy.py", line 27, in handle_client
    received=clientScocket.recv(4096)
OSError: [Errno 9] Bad file descriptor
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Proxy.py
proxy ready
Traceback (most recent call last):
  File "Proxy.py", line 59, in <module>
    handle_client(connectionSocket)
  File "Proxy.py", line 27, in handle_client
    received=clientScocket.recv(4096)
OSError: [Errno 9] Bad file descriptor
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

make a request with another customer at the same time:

<u>Figure</u>6: Two-customer query testing simultaneously

Alas the proxy can't handle both queries simultaneously because it's not configured for**<u>it.</u>**

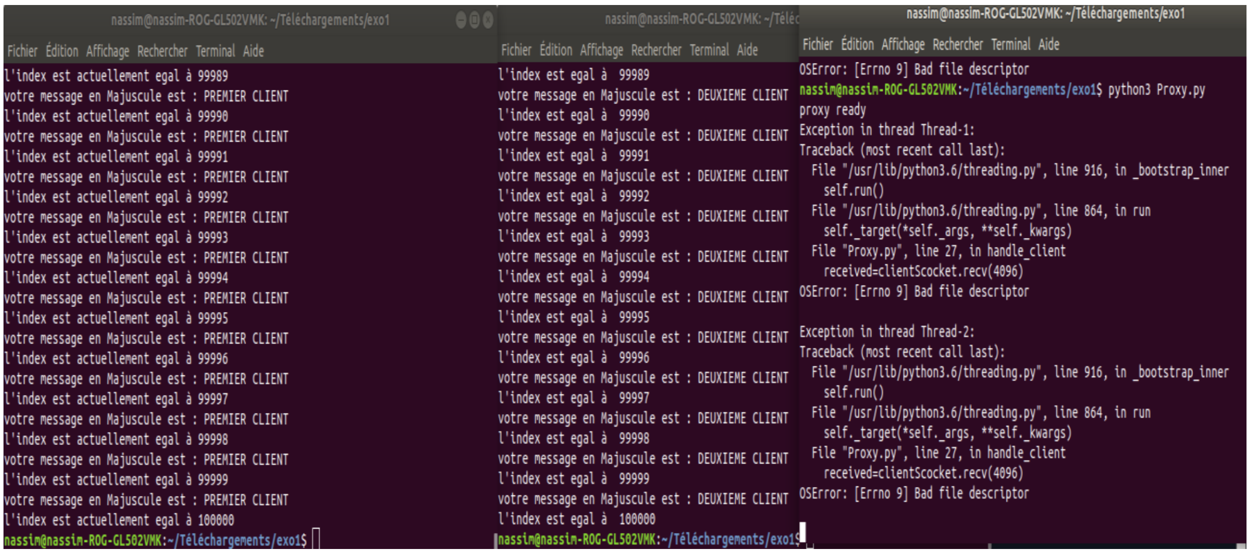
4.Change the Proxy withthreads:

In order for the proxy to respond to several customer queries that arrive at the same time, it needs to create sub-processes that will manage queries and for this we add multi-threading by doing as follows:

```
from threading import *
Thread(target=handle_client,args=(connectionSocket,)).start()
```

- A simple command line allows the proxy to handle multiple customer queries

  This time we try to make queries with two instances of customers on the same machine and the result is satisfactory, the proxy manages to manage queries by opening threads to each customer.


nassim@nassim-ROG-GL502VMK: ~/Téléchargements/exo1
Fichier Édition Affichage Rechercher Terminal Aide
nassim@nassim-ROG-GL502VMK: ~/Téléc
Fichier Édition Affichage Rechercher Terminal Aide
nassim@nassim-ROG-GL502VMK: ~/Téléchargements/exo1
Fichier Édition Affichage Rechercher Terminal Aide

```
l'index est actuellement egal à 99989
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99990
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99991
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99992
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99993
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99994
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99995
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99996
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99997
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99998
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 99999
votre message en Majuscule est : PREMIER CLIENT
l'index est actuellement egal à 100000
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

```
l'index est egal à  99989
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99990
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99991
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99992
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99993
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99994
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99995
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99996
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99997
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99998
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  99999
votre message en Majuscule est : DEUXIEME CLIENT
l'index est egal à  100000
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$
```

```
OSError: [Errno 9] Bad file descriptor
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo1$ python3 Proxy.py
proxy ready
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib/python3.6/threading.py", line 916, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.6/threading.py", line 864, in run
    self._target(*self._args, **self._kwargs)
  File "Proxy.py", line 27, in handle_client
    received=clientSocket.recv(4096)
OSError: [Errno 9] Bad file descriptor

Exception in thread Thread-2:
Traceback (most recent call last):
  File "/usr/lib/python3.6/threading.py", line 916, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.6/threading.py", line 864, in run
    self._target(*self._args, **self._kwargs)
  File "Proxy.py", line 27, in handle_client
    received=clientSocket.recv(4096)
OSError: [Errno 9] Bad file descriptor
```

Figure 7: creating threads for different customer queries

Part 2: Proxy HTTP

In this part, we try to program a proxy entirely dedicated to the HTTP protocol, and therefore all exchanges between the client and the server through the proxy use the HTTP protocol.

1. Creating an HTTP cache

A cache is a folder or file where you can find all the recent searches of Http queries issued by a web client, the cache can contain the path ainsi as the content of the requested file (name of the requested file as well as its contents) whether it is a text, a hyperlink or even images.

To create a cache the program is told in python the following:

- The proxy receives and processes the httpquery.
- If the file is already saved in the cache, it should be opened, read and sent to thecustomer.
- otherwise, you have to pick it up from the server where it is, save it in the cache and send it back to theclient.

The code is as follows:

Study of the 1st car:

```
def Get_from_server( clientSocket,received):
if not received:
clientSocket.close()
else:
proxySocket = socket(AF_INET,SOCK_STREAM)
proxySocket.connect((servername,serverport))
proxySocket.send(received)
Message=proxySocket.recv(4096)
OKreply="HTTP/1.1 200 OK\r\n\r\n"
clientSocket.sendall(OKreply.encode())
while Message:
Message=proxySocket.recv(4096)
clientSocket.send(Message)
proxySocket.close()
clientSocket.close()
```

- A function is set for the case the file is in thecache.
- The sockette receives the request redirectsit to the server socket te to get the FILE of the GET query, from there it returns the contents of the file to the client.

Study of the second case:

Mini-Project 1: Proxy TCP and Proxy HTTP

```
def envoie_Direct(clientSocket,received):
fichier = received.split()[1]
file[1:]
```

<u>Note:</u>The format of an HTTP query is: **GET/index.jpeg HTTP/1.1 Host: 127.0.0.1:1234,** so when you need to get only the name of the requested file you have to separate the query when you find spaces and remove only the first field here "index". "jpeg"

```
try:
with open(fichier,'rb') as f :
OKreply="HTTP/1.1 200 OK\r\n\r\n"
clientSocket.send(OKreply.encode())
file=f.read()
while file:
clientSocket.sendall(file)
file = f.read()
except:
Err="HTTP/1.1 404 File Not Found\r\n\r\n"
clientSocket.send(Err.encode())
clientSocket.close()
```

- In casewhere the requested file is already in thecache.
- The function opens the so-called file, sends a positive response "OKreply".
- It reads the file and sends the content bit by bit without decoding and encoding.
- If it fails to do so, it sends a 404 not found errormessage.

<u>Opening and writing function in the cache:</u>

```
def cache (req) :
file = req.split()[1]
file=str(file[1:].decode())
with open("cache.txt",'r') as f :
file f.read()
look=fichier.find(file) #return -1 in failure
```

- We search for the name of the file obtained in the client'sGAND query in thecache.

```
if look != -1 :
print ("the fichier__"file" __est already recorded in the cache")
k=0
```

- One reasons by the absurd if the find function does not fail is that the file is in the cache.

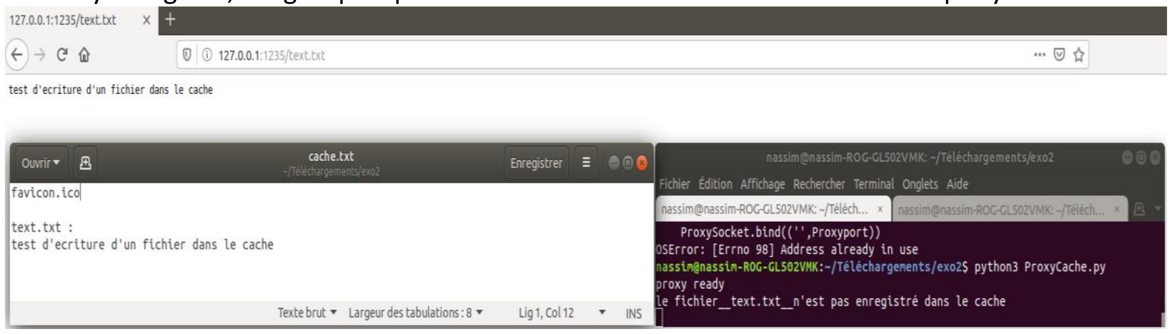
```
else:
        temp=open("cache.txt",'a')
temp2=open(file,"r")
content-temp2.read() #lire the contents of the file ask
temp.write (file) #ecrire file name
temp.write
temp.write (content) #ecrire file content
k=1
temp.close()
temp2.close()
f.close()
return k
```

- If the file is not in the cache, then we open the cache and the file obtained by the server and write the name and contents of the file in thecache.

Inthe end, these functions are grouped into the same function when the proxy becomes a kind of customer:

```
def handle_client(clientSocket):
received = clientSocket.recv(4096)
enreg cache (received) #le requested file is in the cache
If enreg 1:
Get_from_server(clientSocket,received) #Get file from server
else :
envoi_Direct(clientSocket,received) #Get file from cache
clientSocket.close()
```

- By doing this, we group all possible cases when we receive an HTTP GET query



```
127.0.0.1:1235/text.txt

test d'ecriture d'un fichier dans le cache
```

```
cache.txt
~/Téléchargements/exo2

favicon.ico

text.txt :
test d'ecriture d'un fichier dans le cache

Texte brut ▾   Largeur des tabulations : 8 ▾   Lig 1, Col 12   INS
```

```
nassim@nassim-ROG-GL502VMK: ~/Téléchargements/exo2
    ProxySocket.bind(('',Proxyport))
OSError: [Errno 98] Address already in use
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ProxyCache.py
proxy ready
le fichier__text.txt__n'est pas enregistré dans le cache
```

Figure 8: Testing for a file that is not in the cache

2.Creating a Loggeur http

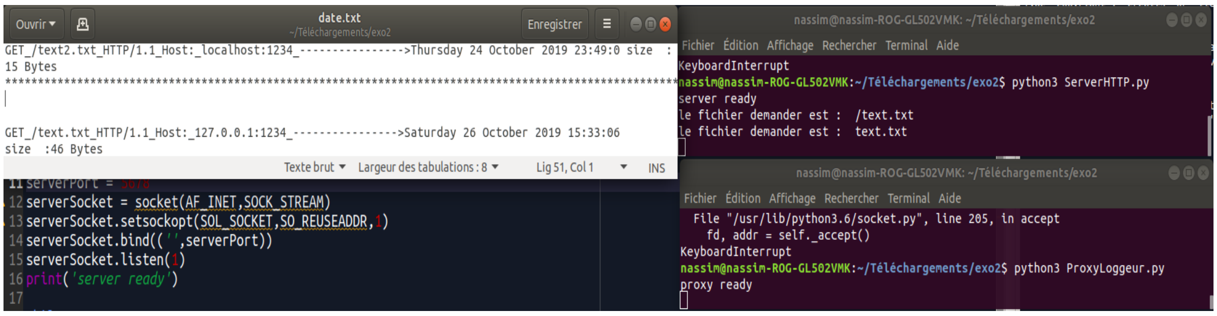
An HTTP loggeur is very simple to make, just record the date at which the query was issued as well as the size of the file obtained during theresponse.
Loggeur Code:

```
def loggeur (received) :
file=received.split()[1]
file=str(file[1:].decode('utf_8'))
log-time.strftime ("%A %d %B %Y %H:%M:%S") #date of the GET requete
received=received.decode('utf_8')
indx=0
r=""
size-str (os.path.getsize (file)) #taille of the file received in response
```

- The date of the query is backed up with a time library function
- backing up the size of the file with a bone library function

```
while indx<5 :
r+=str(received.split()[indx]+"_")
indx=indx+1
f=open("date.txt","a")
f.write(r+"---------------->"+log+"  size  :"+size+" Bytes \n")
f.close()
```

- One writes in a date file.txt word for word the syntax of the GET http query followed by the date at which it was issued and finally the size of the requested file.



```
GET_/text2.txt_HTTP/1.1_Host:_localhost:1234_---------------->Thursday 24 October 2019 23:49:0 size  :
15 Bytes
*****************************************************************************************

GET_/text.txt_HTTP/1.1_Host:_127.0.0.1:1234_---------------->Saturday 26 October 2019 15:33:06
size  :46 Bytes
```

```
serverPort = 5678
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('server ready')
```

```
KeyboardInterrupt
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ServerHTTP.py
server ready
le fichier demander est :  /text.txt
le fichier demander est :  text.txt
```

```
  File "/usr/lib/python3.6/socket.py", line 205, in accept
    fd, addr = self._accept()
KeyboardInterrupt
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ProxyLoggeur.py
proxy ready
```

date.txt
~/Téléchargements/exo2
Ouvrir
Enregistrer
Texte brut
Largeur des tabulations : 8
Lig 51, Col 1
INS
nassim@nassim-ROG-GL502VMK: ~/Téléchargements/exo2
Fichier Édition Affichage Rechercher Terminal Aide

<u>Figure</u> 9: Proxy Loggeur test and file date and size record

<u>3 . Censeur HTTP :</u>

A censor http is a kind of no-fly list for any traveler, except that the traveler here is a file, an image or even a URL of a banned site, this list contains all the names that one wants to ban when receiving an HTTP request.

<u>The Censor's code:</u>

```
def censure (request):
file = request.split()[1]
file=str(file[1:].decode())
with open("censure.txt",'r') as f :
file f.read()
look=fichier.find(file)
```

- We open the censor file.txt and find out if the query file is written in thatlist.

```
if look == -1 :
print ("the file demandé__"file" __n't in the forbidden list))
k=1
```

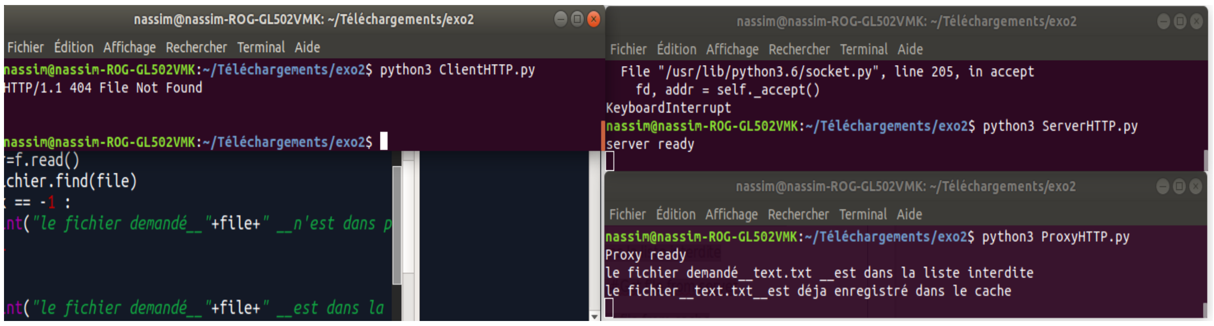
- Si the file is not in this list, we allow the customer sub process of the proxy to do its job (return the file if it is in the cache otherwise it will look for it from the server)

```
else:
print ("the file demandé__ file" __est in the forbidden list)
k=0
f.close()
return k
```

- If the requested file is prohibited, the proxy sous_process is made to return a 404 not founderror.

```
………..text has been ommited …………………..
prohibited censorship (received) #le requested file is in the prohibited list
If prohibited: #le file is not in the prohibited list
If enreg 1:
Get_from_server(clientSocket,received) #Get file from server
else :
envoi_Direct(clientSocket,received) #Get file from cache
else :
Err="HTTP/1.1 404 File Not Found\r\n\r\n"
clientSocket.send(Err.encode())
clientSocket.close()
```

Mini-Project 1: Proxy TCP and Proxy HTTP



```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ClientHTTP.py
HTTP/1.1 404 File Not Found

nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$
=f.read()
chier.find(file)
 == -1 :
nt("le fichier demandé_ "+file+" _n'est dans p

nt("le fichier demandé_ "+file+" _est dans la
```

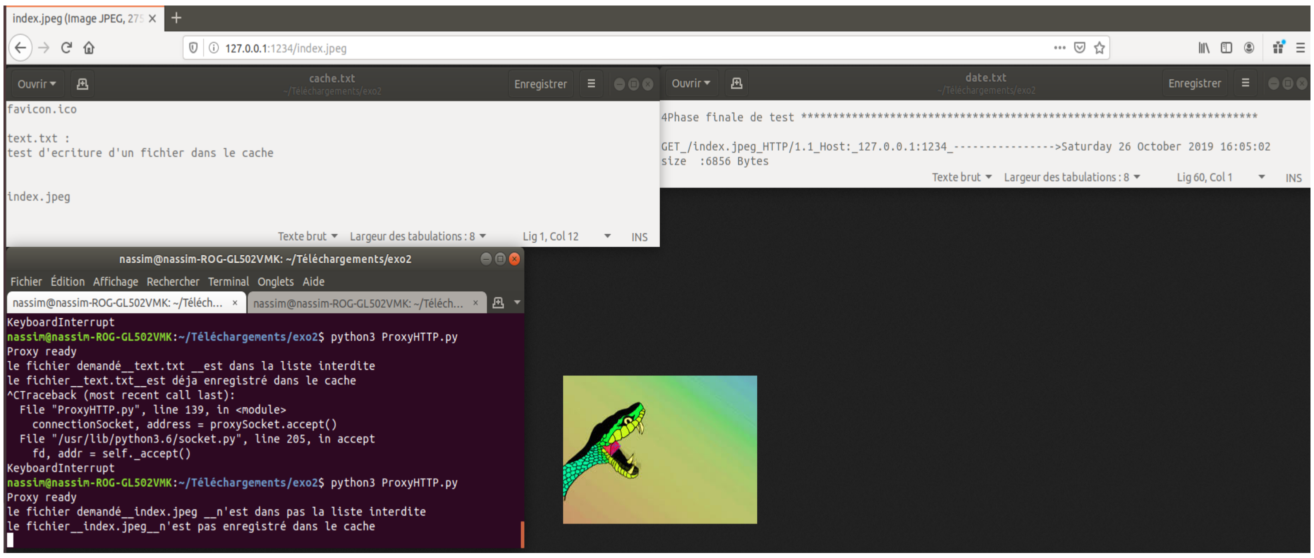
```
  File "/usr/lib/python3.6/socket.py", line 205, in accept
    fd, addr = self._accept()
KeyboardInterrupt
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ServerHTTP.py
server ready
```

```
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ProxyHTTP.py
Proxy ready
le fichier demandé__text.txt __est dans la liste interdite
le fichier__text.txt__est déja enregistré dans le cache
```

Figure 10: Test the censor proxy when the file is in the banned list

4 . Full proxy HTTP test:

The features of the previously defined proxy are followed up:



127.0.0.1:1234/index.jpeg

cache.txt

```
favicon.ico

text.txt :
test d'ecriture d'un fichier dans le cache


index.jpeg
```

date.txt

```
4Phase finale de test *************************************************************************

GET_/index.jpeg_HTTP/1.1_Host:_127.0.0.1:1234_--------------->Saturday 26 October 2019 16:05:02
size  :6856 Bytes
```

```
KeyboardInterrupt
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ProxyHTTP.py
Proxy ready
le fichier demandé__text.txt __est dans la liste interdite
le fichier__text.txt__est déja enregistré dans le cache
^CTraceback (most recent call last):
  File "ProxyHTTP.py", line 139, in <module>
    connectionSocket, address = proxySocket.accept()
  File "/usr/lib/python3.6/socket.py", line 205, in accept
    fd, addr = self._accept()
KeyboardInterrupt
nassim@nassim-ROG-GL502VMK:~/Téléchargements/exo2$ python3 ProxyHTTP.py
Proxy ready
le fichier demandé__index.jpeg __n'est dans pas la liste interdite
le fichier__index.jpeg__n'est pas enregistré dans le cache
```

Figure 11: Tests a proxy with all functions

Conclusion :

A proxy is an essential intermediary in any communication between a server and customers, it does not account for its usefulness when behind a client screen but almost all the queries that you go through the internet goes through a local proxy. It helps to protect us from malicious sites (proxy filtering), makes the response to our daily server queries faster like weather forecast sites that we check before going out morning and evening (proxy cache) and facilitates the administration of the network by engineers (logging queries andhistory).