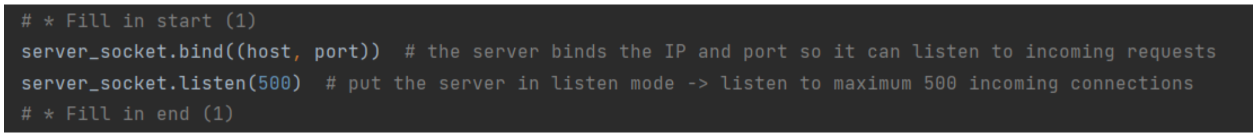
Assignment 2 – Calculator

1.

Server:

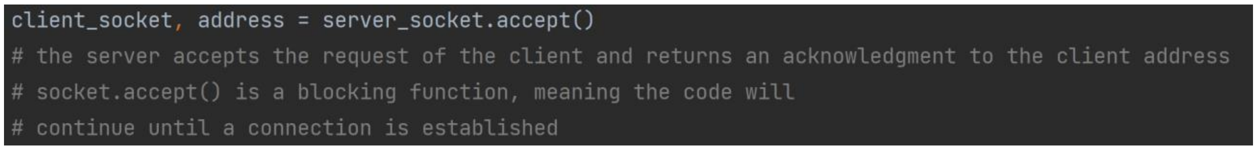
Screenshots:

(1)



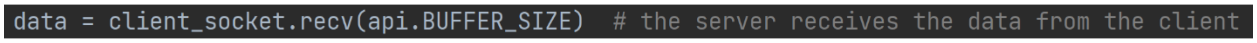
```
# * Fill in start (1)
server_socket.bind((host, port))  # the server binds the IP and port so it can listen to incoming requests
server_socket.listen(500)  # put the server in listen mode -> listen to maximum 500 incoming connections
# * Fill in end (1)
```

(2)



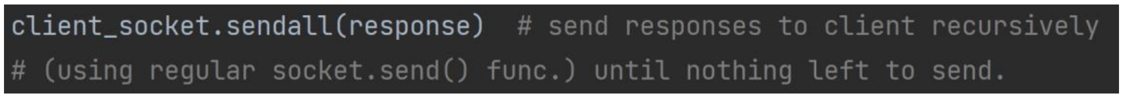
```
client_socket, address = server_socket.accept()
# the server accepts the request of the client and returns an acknowledgment to the client address
# socket.accept() is a blocking function, meaning the code will
# continue until a connection is established
```

(3)



```
data = client_socket.recv(api.BUFFER_SIZE)  # the server receives the data from the client
```

(4)



```
client_socket.sendall(response)  # send responses to client recursively
# (using regular socket.send() func.) until nothing left to send.
```

Explanations:

(1) bind() is method which binds the socket it to a specific IP and port so that it can listen to incoming requests to that port. The listen() method puts the server into listen mode and the argument is the number of connections it could listen to. We don’t want that the proxy will get endless requests so we limited for 500 requests.

(2) accept() is method that used to retrieve a connect request and convert that into a request. The output is a tuple of a socket and IP address. Once the connection is made, we can unpack the tuple into 2 variables for further use (sending responses).

(3) The recv() method reads the data sent to the server, using the default buffer size as the argument.

(4) sendall() this it method to sends all incoming data back by repeatedly

calls send(). So in pic(4) we can see that the client socket get back the response

by the server. And after that it simply closes the client's connection by method

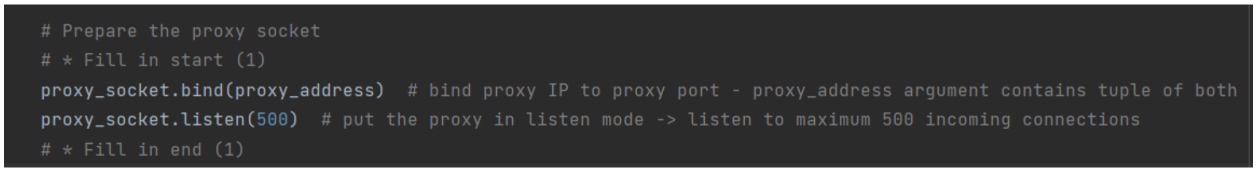
close().

Sum:

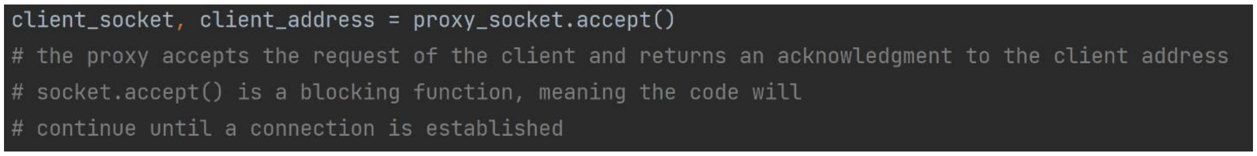
The server socket, binds it to a host and port by bind() method, and start listening for incoming connections by listen() method. To accept an incoming connection we call accept() method which will block until a new client connects. When this happens, it creates a new socket and returns it together with the client's address. Then, in an infinite cycle, it reads data from the socket in batches of 1024 bytes using method recv() until it returns an empty string. After that, it sends all incoming data back using a convenient method sendall(). And after that the client closes the connection.

Proxy:

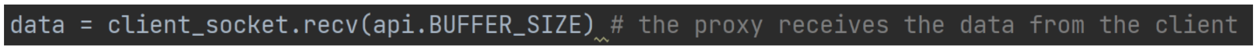
Screenshots:

(1) 

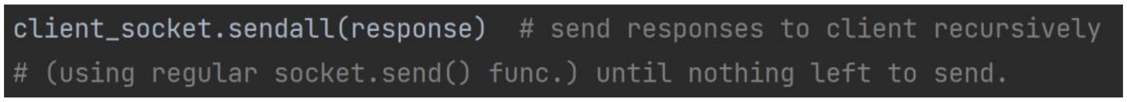
```
# Prepare the proxy socket
# * Fill in start (1)
proxy_socket.bind(proxy_address)  # bind proxy IP to proxy port - proxy_address argument contains tuple of both
proxy_socket.listen(500)  # put the proxy in listen mode -> listen to maximum 500 incoming connections
# * Fill in end (1)
```

(2) 

```
client_socket, client_address = proxy_socket.accept()
# the proxy accepts the request of the client and returns an acknowledgment to the client address
# socket.accept() is a blocking function, meaning the code will
# continue until a connection is established
```

(3) 

```
data = client_socket.recv(api.BUFFER_SIZE) # the proxy receives the data from the client
```

(4) 

```
client_socket.sendall(response)  # send responses to client recursively
# (using regular socket.send() func.) until nothing left to send.
```

Explanations:

(1) The proxy server binds the socket with the default proxy host and IP, using the bind() method as explained above. The listen() method puts the proxy into listen mode and the argument is the number of connections it could listen to. We don’t want that the proxy will get endless requests so we limited for 500 requests.

(2) In the proxy server, we unpack the tuple returned from the accept() method as explained above about the "main" server.

(3) The recv() method reads the data sent to the proxy, using the default buffer size as the argument.

(4) The response is built as follows: if there is a cache "hit", meaning the query was recently sent and its response is in the caches, then the response variable would take its value straight from the cache. Otherwise (cache "miss"), the proxy would acquire a connection with the server send it the query, the server would then save the response in the cache as well as send it back to the proxy which would save it in its response variable. We then send the response to the client via sendall() method.

See diagram below showing the 2 processes explained above.

<u>Sum:</u>

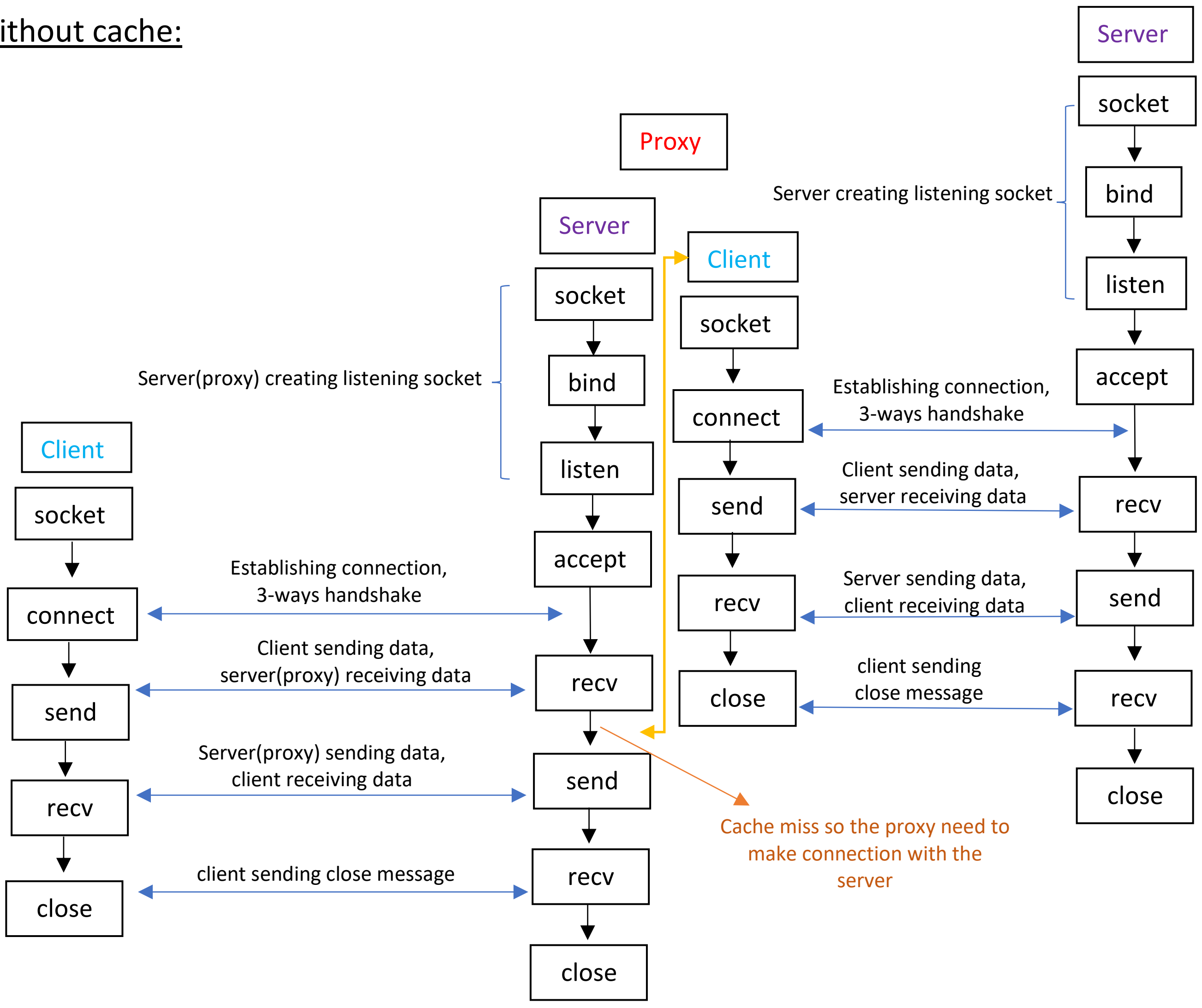
The server proxy, binds it to a proxy_host and proxy_port by bind() method, and start listening for incoming connections by listen() method. To accept an incoming connection we call accept() method which will block until a new client connects. When this happens, it creates a new socket and returns it together with the client's address. Then, if there is a cache "hit", meaning the query was recently sent and its response is in the caches, then the response variable would take its value straight from the cache. Otherwise (cache "miss"), the proxy would acquire a connection with the server send it the query, the server would then save the response in the cache as well as send it back to the proxy which would save it in its response variable. We then send the response to the client via sendall() method. And after that the client closes the connection.

See diagram below showing the 2 processes explained above.

2.

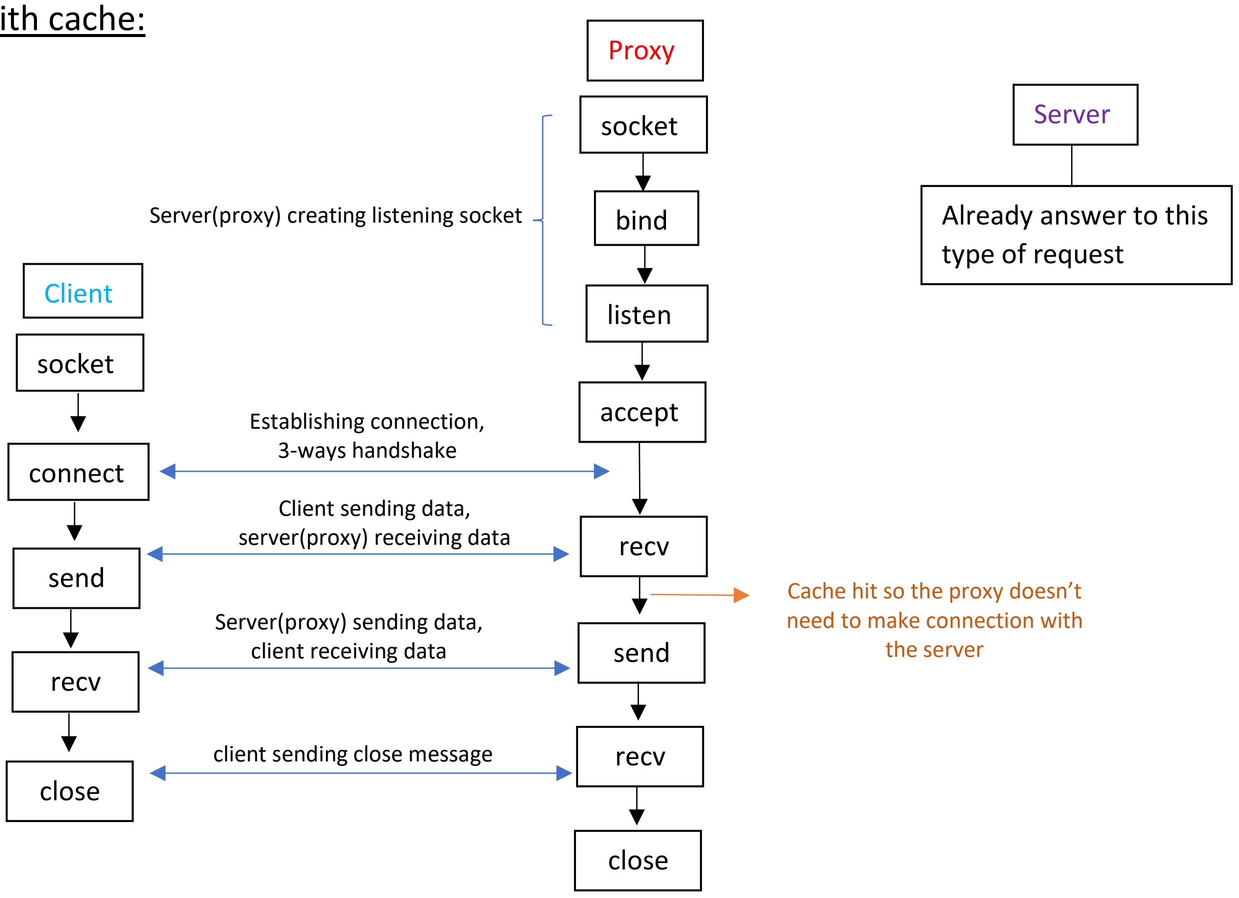
<u>Proxy flowchart:</u>

<u>Without cache:</u>


Server
socket
bind
listen
accept
recv
send
recv
close
Server creating listening socket
Proxy
Server
socket
bind
listen
accept
recv
send
recv
close
Server(proxy) creating listening socket
Client
socket
connect
send
recv
close
Establishing connection, 3-ways handshake
Client sending data, server receiving data
Server sending data, client receiving data
client sending close message
Client
socket
connect
send
recv
close
Establishing connection, 3-ways handshake
Client sending data, server(proxy) receiving data
Server(proxy) sending data, client receiving data
client sending close message
Cache miss so the proxy need to make connection with the server

This flowchart describes what happened when the proxy is listening and the client open request connection when the cache is not hit in the proxy.
when the client opens a request the client is the client, and the proxy is the server. If the proxy doesn’t have the cache, he need to send request connection to the server and now the proxy is the client and the server is server.
When the proxy get the response data from the server and close the connection with the server he send back the response with the data back to the client and then close the connection.

With cache:


Proxy
socket
bind
listen
accept
recv
send
recv
close
Server(proxy) creating listening socket
Server
Already answer to this type of request
Client
socket
connect
send
recv
close
Establishing connection, 3-ways handshake
Client sending data, server(proxy) receiving data
Server(proxy) sending data, client receiving data
client sending close message
Cache hit so the proxy doesn't need to make connection with the server

This flowchart describes what happened when the proxy is listening and the client send request connection when the cache is hit in the proxy.

when the client open a request the client is the client and the proxy is the server. Because the proxy has the cache he don't need to send request connection to the server. So, he can response data from the cache back to the client and then close the connection.

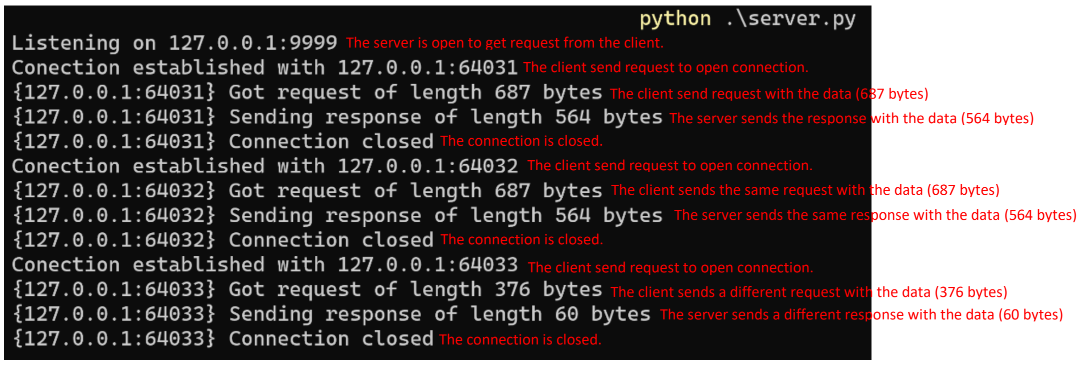
Wire shark:

3.1.

Screenshots:

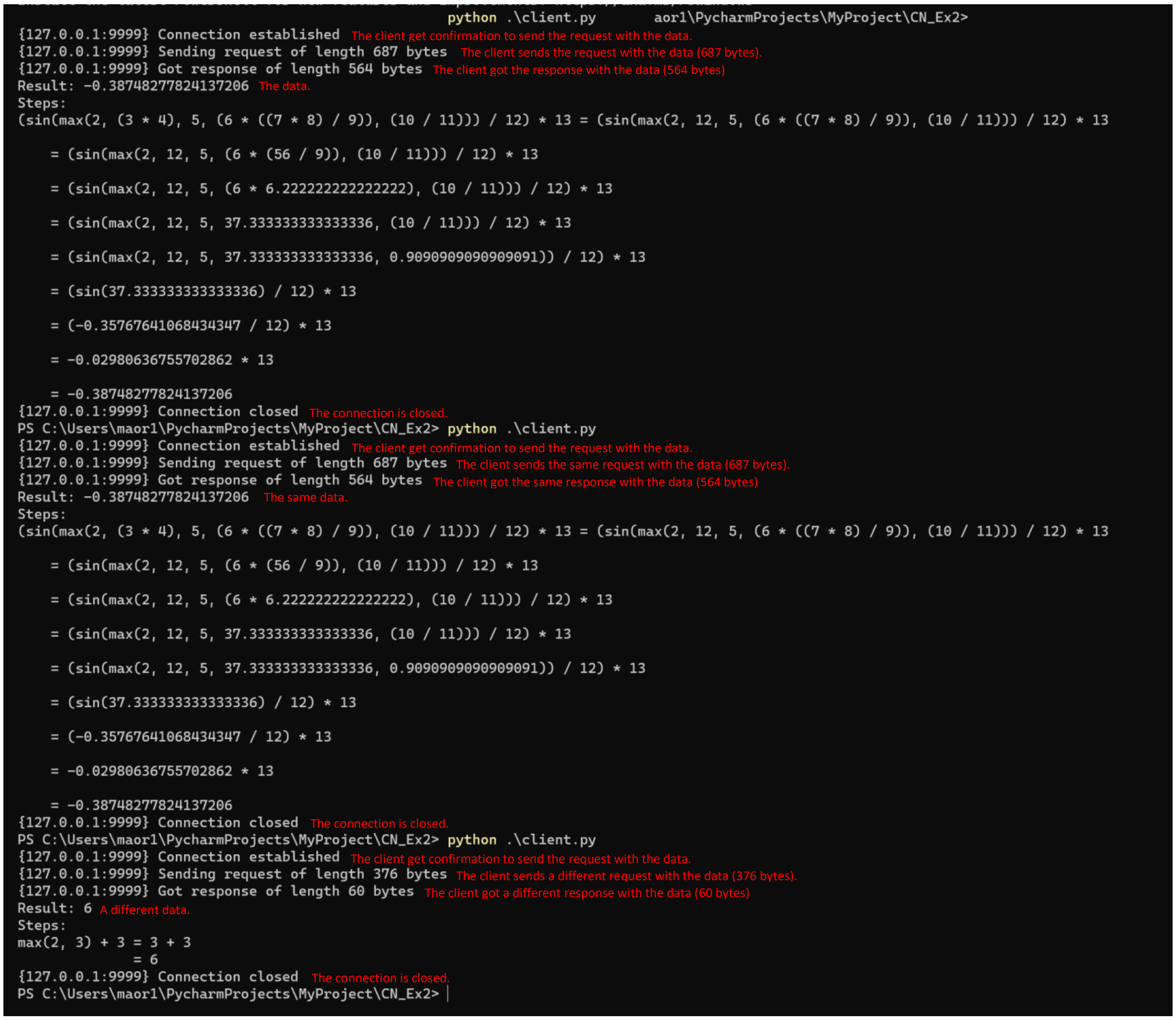
Cmd: Note that our notes are in red – we can see the whole 3 request.

The server:



```
                                                    python .\server.py
Listening on 127.0.0.1:9999 The server is open to get request from the client.
Conection established with 127.0.0.1:64031 The client send request to open connection.
{127.0.0.1:64031} Got request of length 687 bytes The client send request with the data (687 bytes)
{127.0.0.1:64031} Sending response of length 564 bytes The server sends the response with the data (564 bytes)
{127.0.0.1:64031} Connection closed The connection is closed.
Conection established with 127.0.0.1:64032 The client send request to open connection.
{127.0.0.1:64032} Got request of length 687 bytes The client sends the same request with the data (687 bytes)
{127.0.0.1:64032} Sending response of length 564 bytes  The server sends the same response with the data (564 bytes)
{127.0.0.1:64032} Connection closed The connection is closed.
Conection established with 127.0.0.1:64033 The client send request to open connection.
{127.0.0.1:64033} Got request of length 376 bytes The client sends a different request with the data (376 bytes)
{127.0.0.1:64033} Sending response of length 60 bytes The server sends a different response with the data (60 bytes)
{127.0.0.1:64033} Connection closed The connection is closed.
```

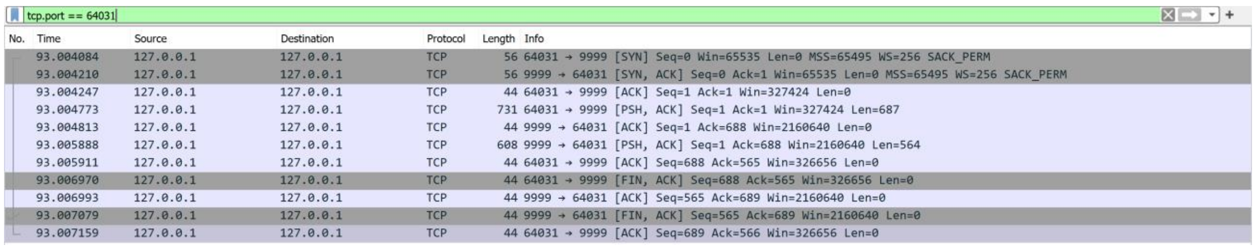
The client:



```
                                                     python .\client.py       aor1\PycharmProjects\MyProject\CN_Ex2>
{127.0.0.1:9999} Connection established  The client get confirmation to send the request with the data.
{127.0.0.1:9999} Sending request of length 687 bytes  The client sends the request with the data (687 bytes).
{127.0.0.1:9999} Got response of length 564 bytes  The client got the response with the data (564 bytes)
Result: -0.38748277824137206  The data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.02980636755702862 * 13

    = -0.38748277824137206
{127.0.0.1:9999} Connection closed  The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2> python .\client.py
{127.0.0.1:9999} Connection established  The client get confirmation to send the request with the data.
{127.0.0.1:9999} Sending request of length 687 bytes  The client sends the same request with the data (687 bytes).
{127.0.0.1:9999} Got response of length 564 bytes  The client got the same response with the data (564 bytes)
Result: -0.38748277824137206  The same data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.02980636755702862 * 13

    = -0.38748277824137206
{127.0.0.1:9999} Connection closed  The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2> python .\client.py
{127.0.0.1:9999} Connection established  The client get confirmation to send the request with the data.
{127.0.0.1:9999} Sending request of length 376 bytes  The client sends a different request with the data (376 bytes).
{127.0.0.1:9999} Got response of length 60 bytes  The client got a different response with the data (60 bytes)
Result: 6  A different data.
Steps:
max(2, 3) + 3 = 3 + 3
              = 6
{127.0.0.1:9999} Connection closed  The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2> |
```

Wire shark:

(1) First request: client -> 64031, server -> 9999



```
tcp.port == 64031
No. Time        Source      Destination   Protocol  Length Info
    93.004084   127.0.0.1   127.0.0.1     TCP         56 64031 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    93.004210   127.0.0.1   127.0.0.1     TCP         56 9999 → 64031 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    93.004247   127.0.0.1   127.0.0.1     TCP         44 64031 → 9999 [ACK] Seq=1 Ack=1 Win=327424 Len=0
    93.004773   127.0.0.1   127.0.0.1     TCP        731 64031 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=687
    93.004813   127.0.0.1   127.0.0.1     TCP         44 9999 → 64031 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
    93.005888   127.0.0.1   127.0.0.1     TCP        608 9999 → 64031 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
    93.005911   127.0.0.1   127.0.0.1     TCP         44 64031 → 9999 [ACK] Seq=688 Ack=565 Win=326656 Len=0
    93.006970   127.0.0.1   127.0.0.1     TCP         44 64031 → 9999 [FIN, ACK] Seq=688 Ack=565 Win=326656 Len=0
    93.006993   127.0.0.1   127.0.0.1     TCP         44 9999 → 64031 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
    93.007079   127.0.0.1   127.0.0.1     TCP         44 9999 → 64031 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
    93.007159   127.0.0.1   127.0.0.1     TCP         44 64031 → 9999 [ACK] Seq=689 Ack=566 Win=326656 Len=0
```

"3 way handshake":

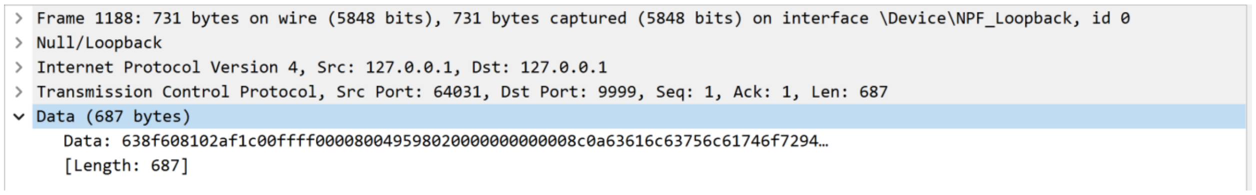
1. client sends SYN request – a request to connect with server.

2. Server sends ACK and SYN to connect with client.

3. Client sends ACK to approve connection

Sending query:

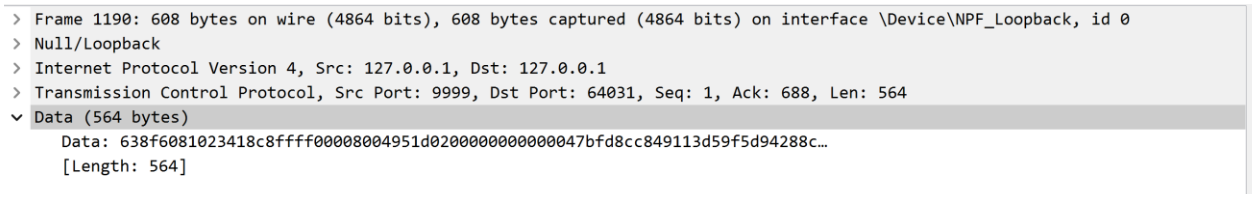
4. Client sends PSH package – query to server



```
> Frame 1188: 731 bytes on wire (5848 bits), 731 bytes captured (5848 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 64031, Dst Port: 9999, Seq: 1, Ack: 1, Len: 687
v Data (687 bytes)
    Data: 638f608102af1c00ffff0000800495980200000000000008c0a63616c63756c61746f7294…
    [Length: 687]
```

5. Server sends ACK acknowledging receiving the query

6. Server sends ACK + PSH – response to client.



```
> Frame 1190: 608 bytes on wire (4864 bits), 608 bytes captured (4864 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 9999, Dst Port: 64031, Seq: 1, Ack: 688, Len: 564
v Data (564 bytes)
    Data: 638f6081023418c8ffff00008004951d0200000000000047bfd8cc849113d59f5d94288c…
    [Length: 564]
```

7. Client send ACK acknowledging receiving response.

Closing connection:

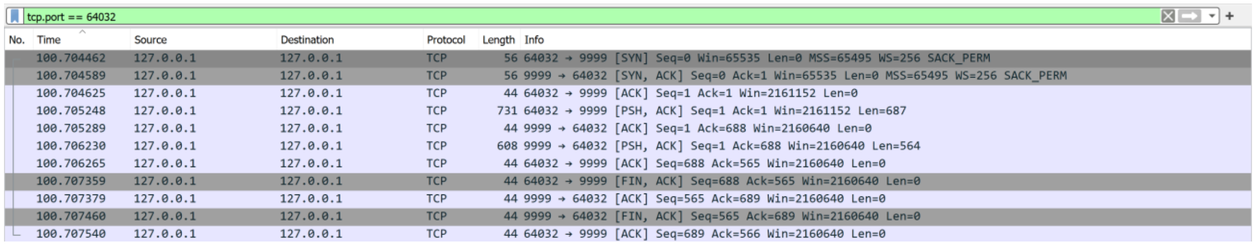
8. Server sends package requesting to end connection – "FIN"

9. Client acknowledges

10.Client sends request to end connection with server.

11. Server acknowledges.

(2) Second request(same one): client -> 64032, server -> 9999



```
tcp.port == 64032
No. Time        Source      Destination  Protocol Length Info
    100.704462  127.0.0.1   127.0.0.1    TCP       56 64032 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    100.704589  127.0.0.1   127.0.0.1    TCP       56 9999 → 64032 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    100.704625  127.0.0.1   127.0.0.1    TCP       44 64032 → 9999 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
    100.705248  127.0.0.1   127.0.0.1    TCP      731 64032 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=687
    100.705289  127.0.0.1   127.0.0.1    TCP       44 9999 → 64032 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
    100.706230  127.0.0.1   127.0.0.1    TCP      608 9999 → 64032 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
    100.706265  127.0.0.1   127.0.0.1    TCP       44 64032 → 9999 [ACK] Seq=688 Ack=565 Win=2160640 Len=0
    100.707359  127.0.0.1   127.0.0.1    TCP       44 64032 → 9999 [FIN, ACK] Seq=688 Ack=565 Win=2160640 Len=0
    100.707379  127.0.0.1   127.0.0.1    TCP       44 9999 → 64032 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
    100.707460  127.0.0.1   127.0.0.1    TCP       44 9999 → 64032 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
    100.707540  127.0.0.1   127.0.0.1    TCP       44 64032 → 9999 [ACK] Seq=689 Ack=566 Win=2160640 Len=0
```

"3 way handshake":

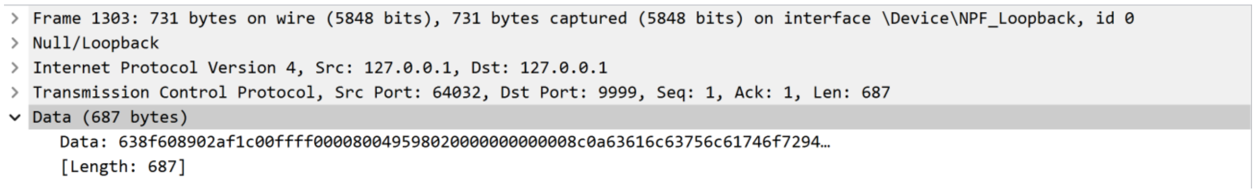
1. client sends SYN request – a request to connect with server.

2. Server sends ACK and SYN to connect with client.

3. Client sends ACK to approve connection

Sending query:

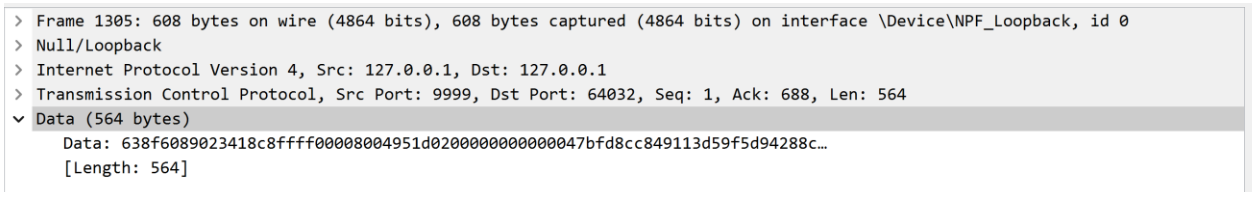
4. Client sends PSH package – query to server we can see that is the same data.



```
> Frame 1303: 731 bytes on wire (5848 bits), 731 bytes captured (5848 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 64032, Dst Port: 9999, Seq: 1, Ack: 1, Len: 687
v Data (687 bytes)
    Data: 638f608902af1c00ffff00008004959802000000000008c0a63616c63756c61746f7294…
    [Length: 687]
```

5. Server sends ACK acknowledging receiving the query

6. Server sends ACK + PSH – response to client. Again we can see that is the same response data.



```
> Frame 1305: 608 bytes on wire (4864 bits), 608 bytes captured (4864 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 9999, Dst Port: 64032, Seq: 1, Ack: 688, Len: 564
v Data (564 bytes)
    Data: 638f6089023418c8ffff00008004951d0200000000000047bfd8cc849113d59f5d94288c…
    [Length: 564]
```

7. Client send ACK acknowledging receiving response.

Closing connection:

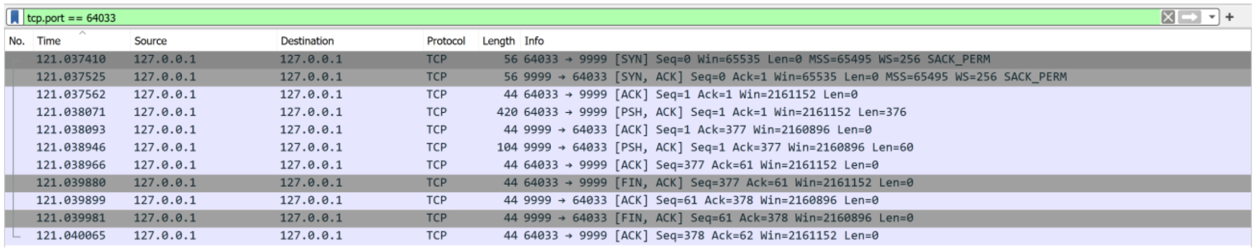
8. Server sends package requesting to end connection – "FIN"

9. Client acknowledges

10.Client sends request to end connection with server.

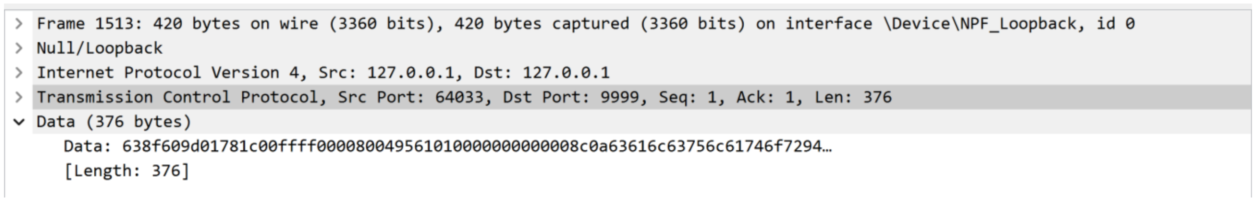
11. Server acknowledges.

(3) Third request(diffrenet one): client -> 64033, server -> 9999



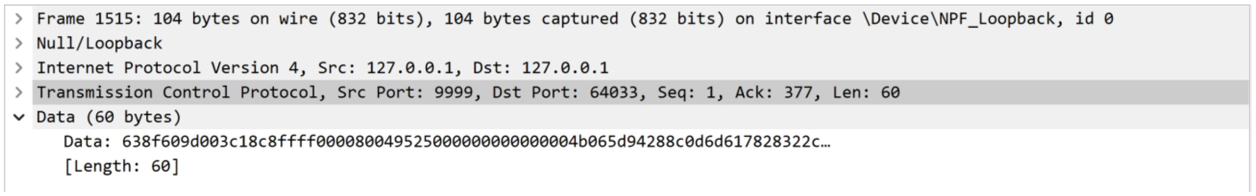
```
tcp.port == 64033
No. Time         Source      Destination  Protocol  Length Info
    121.037410   127.0.0.1   127.0.0.1    TCP           56 64033 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    121.037525   127.0.0.1   127.0.0.1    TCP           56 9999 → 64033 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    121.037562   127.0.0.1   127.0.0.1    TCP           44 64033 → 9999 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
    121.038071   127.0.0.1   127.0.0.1    TCP          420 64033 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=376
    121.038093   127.0.0.1   127.0.0.1    TCP           44 9999 → 64033 [ACK] Seq=1 Ack=377 Win=2160896 Len=0
    121.038946   127.0.0.1   127.0.0.1    TCP          104 9999 → 64033 [PSH, ACK] Seq=1 Ack=377 Win=2160896 Len=60
    121.038966   127.0.0.1   127.0.0.1    TCP           44 64033 → 9999 [ACK] Seq=377 Ack=61 Win=2161152 Len=0
    121.039880   127.0.0.1   127.0.0.1    TCP           44 64033 → 9999 [FIN, ACK] Seq=377 Ack=61 Win=2161152 Len=0
    121.039899   127.0.0.1   127.0.0.1    TCP           44 9999 → 64033 [ACK] Seq=61 Ack=378 Win=2160896 Len=0
    121.039981   127.0.0.1   127.0.0.1    TCP           44 9999 → 64033 [FIN, ACK] Seq=61 Ack=378 Win=2160896 Len=0
    121.040065   127.0.0.1   127.0.0.1    TCP           44 64033 → 9999 [ACK] Seq=378 Ack=62 Win=2161152 Len=0
```

Here we send a different query. The process is the same except for the sizes (len – 375) of the query sent from the client.



```
> Frame 1513: 420 bytes on wire (3360 bits), 420 bytes captured (3360 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 64033, Dst Port: 9999, Seq: 1, Ack: 1, Len: 376
v Data (376 bytes)
    Data: 638f609d01781c00ffff000080049561010000000000008c0a63616c63756c61746f7294…
    [Length: 376]
```

We can see that the response sent back from the server is different (len – 60)



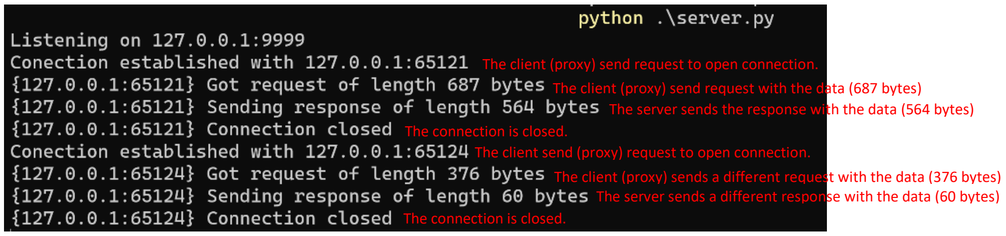
```
> Frame 1515: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 9999, Dst Port: 64033, Seq: 1, Ack: 377, Len: 60
v Data (60 bytes)
    Data: 638f609d003c18c8ffff0000800495250000000000000004b065d94288c0d6d617828322c…
    [Length: 60]
```

3.2.

<u>Screenshots:</u>

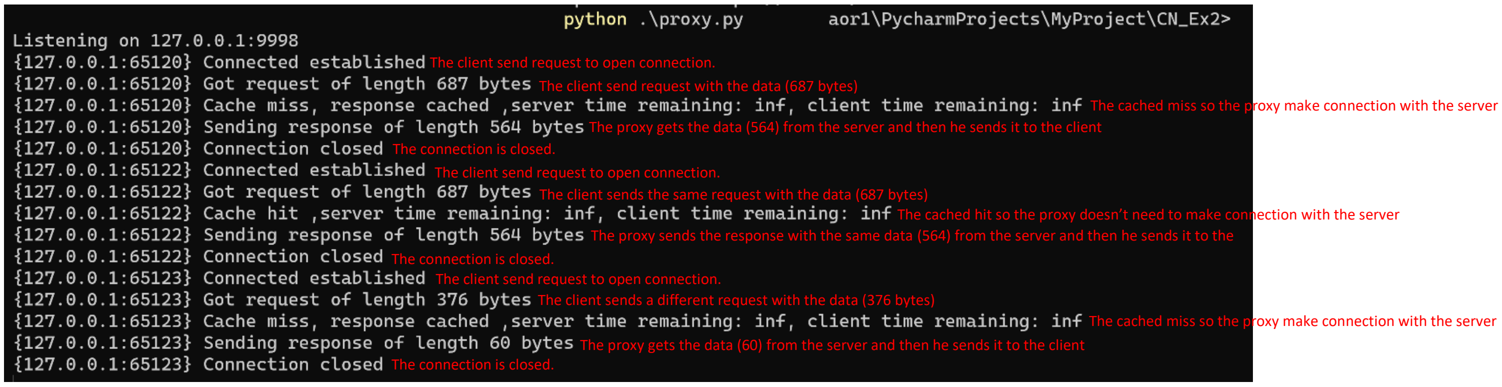
Cmd: Note that our notes are in red.

The server: We can see that the server get two request because the proxy is on and when the client ask the proxy, if the cache miss in the proxy he had to get connected with the server and when he had the data in cache he didn't had to get connected with the server.



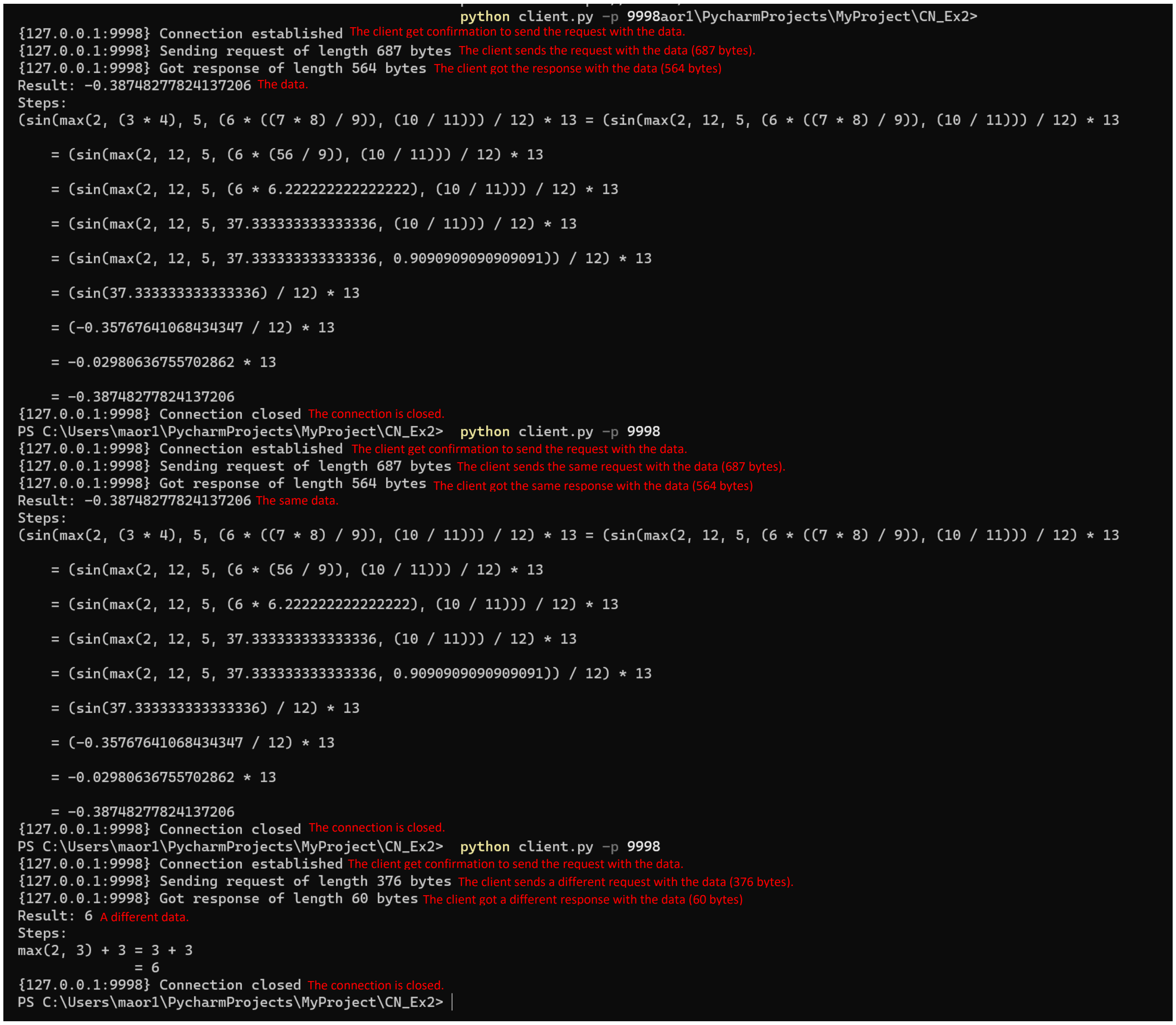
```
                                                            python .\server.py
Listening on 127.0.0.1:9999
Conection established with 127.0.0.1:65121 The client (proxy) send request to open connection.
{127.0.0.1:65121} Got request of length 687 bytes The client (proxy) send request with the data (687 bytes)
{127.0.0.1:65121} Sending response of length 564 bytes The server sends the response with the data (564 bytes)
{127.0.0.1:65121} Connection closed The connection is closed.
Conection established with 127.0.0.1:65124 The client send (proxy) request to open connection.
{127.0.0.1:65124} Got request of length 376 bytes The client (proxy) sends a different request with the data (376 bytes)
{127.0.0.1:65124} Sending response of length 60 bytes The server sends a different response with the data (60 bytes)
{127.0.0.1:65124} Connection closed The connection is closed.
```

The proxy:



```
                                                    python .\proxy.py        aor1\PycharmProjects\MyProject\CN_Ex2>
Listening on 127.0.0.1:9998
{127.0.0.1:65120} Connected established The client send request to open connection.
{127.0.0.1:65120} Got request of length 687 bytes The client send request with the data (687 bytes)
{127.0.0.1:65120} Cache miss, response cached ,server time remaining: inf, client time remaining: inf The cached miss so the proxy make connection with the server
{127.0.0.1:65120} Sending response of length 564 bytes The proxy gets the data (564) from the server and then he sends it to the client
{127.0.0.1:65120} Connection closed The connection is closed.
{127.0.0.1:65122} Connected established The client send request to open connection.
{127.0.0.1:65122} Got request of length 687 bytes The client sends the same request with the data (687 bytes)
{127.0.0.1:65122} Cache hit ,server time remaining: inf, client time remaining: inf The cached hit so the proxy doesn't need to make connection with the server
{127.0.0.1:65122} Sending response of length 564 bytes The proxy sends the response with the same data (564) from the server and then he sends it to the
{127.0.0.1:65122} Connection closed The connection is closed.
{127.0.0.1:65123} Connected established The client send request to open connection.
{127.0.0.1:65123} Got request of length 376 bytes The client sends a different request with the data (376 bytes)
{127.0.0.1:65123} Cache miss, response cached ,server time remaining: inf, client time remaining: inf The cached miss so the proxy make connection with the server
{127.0.0.1:65123} Sending response of length 60 bytes The proxy gets the data (60) from the server and then he sends it to the client
{127.0.0.1:65123} Connection closed The connection is closed.
```

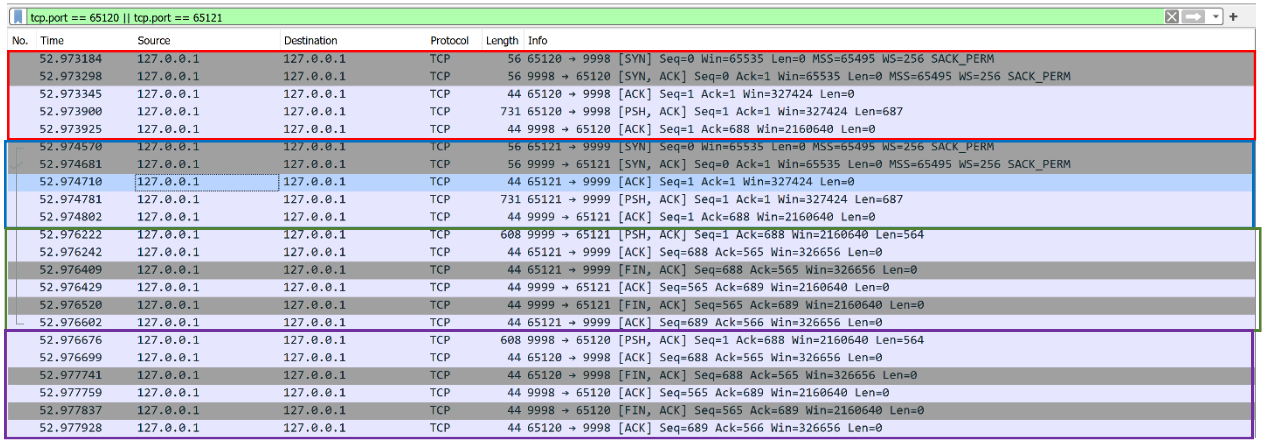
The client: here the client is the client, and the server is the proxy.



```
                                                     python client.py -p 9998aor1\PycharmProjects\MyProject\CN_Ex2>
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 687 bytes The client sends the request with the data (687 bytes).
{127.0.0.1:9998} Got response of length 564 bytes The client got the response with the data (564 bytes)
Result: -0.38748277824137206 The data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.029806367557028620 * 13

    = -0.38748277824137206
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2>  python client.py -p 9998
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 687 bytes The client sends the same request with the data (687 bytes).
{127.0.0.1:9998} Got response of length 564 bytes The client got the same response with the data (564 bytes)
Result: -0.38748277824137206 The same data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.029806367557028620 * 13

    = -0.38748277824137206
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2>  python client.py -p 9998
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 376 bytes The client sends a different request with the data (376 bytes).
{127.0.0.1:9998} Got response of length 60 bytes The client got a different response with the data (60 bytes)
Result: 6 A different data.
Steps:
max(2, 3) + 3 = 3 + 3
              = 6
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2> 
```

(1) First request: client ->65120, server(proxy) ->9998

client(proxy) ->65121, server ->9999



```
tcp.port == 65120 || tcp.port == 65121
No.  Time        Source      Destination  Protocol  Length  Info
     52.973184   127.0.0.1   127.0.0.1    TCP        56 65120 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     52.973298   127.0.0.1   127.0.0.1    TCP        56 9998 → 65120 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     52.973345   127.0.0.1   127.0.0.1    TCP        44 65120 → 9998 [ACK] Seq=1 Ack=1 Win=327424 Len=0
     52.973900   127.0.0.1   127.0.0.1    TCP       731 65120 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=687
     52.973925   127.0.0.1   127.0.0.1    TCP        44 9998 → 65120 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
     52.974570   127.0.0.1   127.0.0.1    TCP        56 65121 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     52.974681   127.0.0.1   127.0.0.1    TCP        56 9999 → 65121 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     52.974710   127.0.0.1   127.0.0.1    TCP        44 65121 → 9999 [ACK] Seq=1 Ack=1 Win=327424 Len=0
     52.974781   127.0.0.1   127.0.0.1    TCP       731 65121 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=687
     52.974802   127.0.0.1   127.0.0.1    TCP        44 9999 → 65121 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
     52.976222   127.0.0.1   127.0.0.1    TCP       608 9999 → 65121 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
     52.976242   127.0.0.1   127.0.0.1    TCP        44 65121 → 9999 [ACK] Seq=688 Ack=565 Win=326656 Len=0
     52.976409   127.0.0.1   127.0.0.1    TCP        44 65121 → 9999 [FIN, ACK] Seq=688 Ack=565 Win=326656 Len=0
     52.976429   127.0.0.1   127.0.0.1    TCP        44 9999 → 65121 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
     52.976520   127.0.0.1   127.0.0.1    TCP        44 9999 → 65121 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
     52.976602   127.0.0.1   127.0.0.1    TCP        44 65121 → 9999 [ACK] Seq=689 Ack=566 Win=326656 Len=0
     52.976676   127.0.0.1   127.0.0.1    TCP       608 9998 → 65120 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
     52.976699   127.0.0.1   127.0.0.1    TCP        44 65120 → 9998 [ACK] Seq=688 Ack=565 Win=326656 Len=0
     52.977741   127.0.0.1   127.0.0.1    TCP        44 65120 → 9998 [FIN, ACK] Seq=688 Ack=565 Win=326656 Len=0
     52.977759   127.0.0.1   127.0.0.1    TCP        44 9998 → 65120 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
     52.977837   127.0.0.1   127.0.0.1    TCP        44 9998 → 65120 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
     52.977928   127.0.0.1   127.0.0.1    TCP        44 65120 → 9998 [ACK] Seq=689 Ack=566 Win=326656 Len=0
```

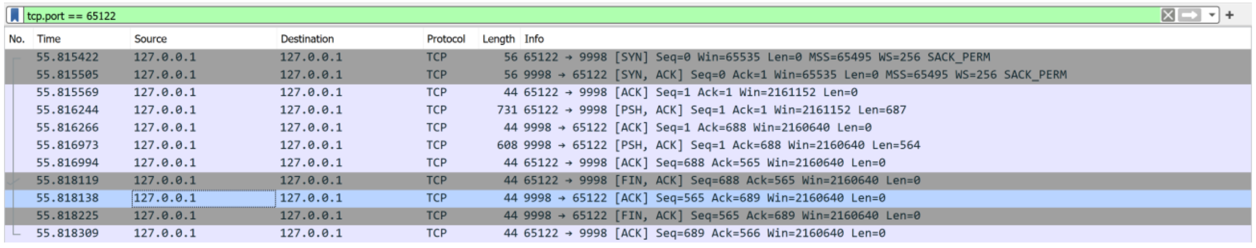
Here the client sends the query to the proxy server after establishing connection in the same way as in question 3.1.

The proxy sends the query to the main server, after establishing connection, because there was a cache "miss" (first time query).

The server sends the response to the proxy and requests end of connection.

Then the proxy sends the response to the client and requests end of connection.

(2) Second request: client ->65122, server(proxy) ->9998



```
tcp.port == 65122
No.  Time        Source      Destination  Protocol  Length  Info
     55.815422   127.0.0.1   127.0.0.1    TCP        56 65122 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     55.815505   127.0.0.1   127.0.0.1    TCP        56 9998 → 65122 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     55.815569   127.0.0.1   127.0.0.1    TCP        44 65122 → 9998 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
     55.816244   127.0.0.1   127.0.0.1    TCP       731 65122 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=687
     55.816266   127.0.0.1   127.0.0.1    TCP        44 9998 → 65122 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
     55.816973   127.0.0.1   127.0.0.1    TCP       608 9998 → 65122 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
     55.816994   127.0.0.1   127.0.0.1    TCP        44 65122 → 9998 [ACK] Seq=688 Ack=565 Win=2160640 Len=0
     55.818119   127.0.0.1   127.0.0.1    TCP        44 65122 → 9998 [FIN, ACK] Seq=688 Ack=565 Win=2160640 Len=0
     55.818138   127.0.0.1   127.0.0.1    TCP        44 9998 → 65122 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
     55.818225   127.0.0.1   127.0.0.1    TCP        44 9998 → 65122 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
     55.818309   127.0.0.1   127.0.0.1    TCP        44 65122 → 9998 [ACK] Seq=689 Ack=566 Win=2160640 Len=0
```

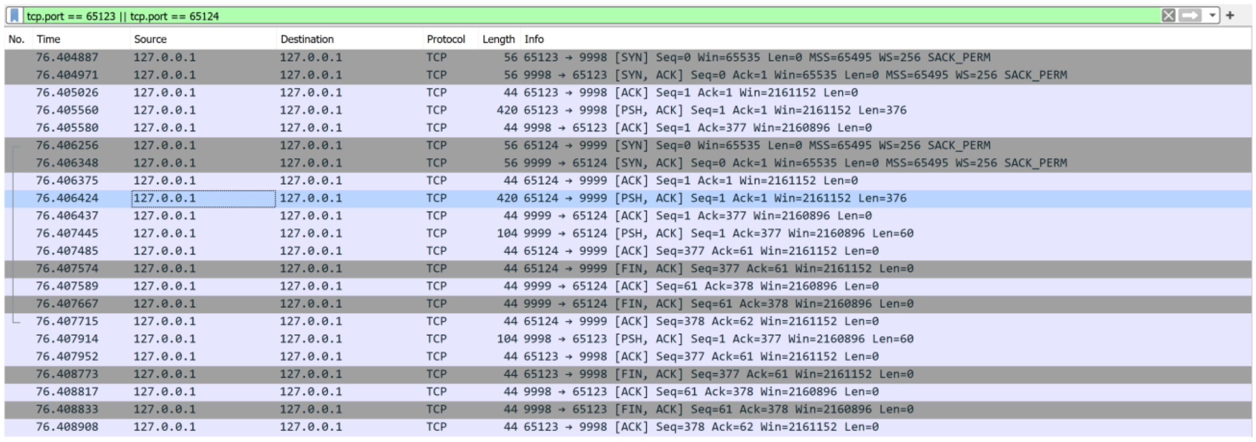
Here the process is the same except for the fact that no connection is established between the proxy and the server side, for it is not necessary, because the proxy "found" the response for the query (which is identical to the last one) int the cache,

meaning there was a cache hit, and there was no need to recalculate the result.

Therefore, the only connection needed to respond to the query was between the client side and the proxy side.

(3) Third request: client ->65123, server(proxy) ->9998

client(proxy) ->65124 , server ->9999



```
tcp.port == 65123 || tcp.port == 65124
No.  Time       Source      Destination  Protocol  Length Info
     76.404887  127.0.0.1   127.0.0.1    TCP           56 65123 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     76.404971  127.0.0.1   127.0.0.1    TCP           56 9998 → 65123 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     76.405026  127.0.0.1   127.0.0.1    TCP           44 65123 → 9998 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
     76.405560  127.0.0.1   127.0.0.1    TCP          420 65123 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=376
     76.405580  127.0.0.1   127.0.0.1    TCP           44 9998 → 65123 [ACK] Seq=1 Ack=377 Win=2160896 Len=0
     76.406256  127.0.0.1   127.0.0.1    TCP           56 65124 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     76.406348  127.0.0.1   127.0.0.1    TCP           56 9999 → 65124 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
     76.406375  127.0.0.1   127.0.0.1    TCP           44 65124 → 9999 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
     76.406424  127.0.0.1   127.0.0.1    TCP          420 65124 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=376
     76.406437  127.0.0.1   127.0.0.1    TCP           44 9999 → 65124 [ACK] Seq=1 Ack=377 Win=2160896 Len=0
     76.407445  127.0.0.1   127.0.0.1    TCP          104 9999 → 65124 [PSH, ACK] Seq=1 Ack=377 Win=2160896 Len=60
     76.407485  127.0.0.1   127.0.0.1    TCP           44 65124 → 9999 [ACK] Seq=377 Ack=61 Win=2161152 Len=0
     76.407574  127.0.0.1   127.0.0.1    TCP           44 65124 → 9999 [FIN, ACK] Seq=377 Ack=61 Win=2161152 Len=0
     76.407589  127.0.0.1   127.0.0.1    TCP           44 9999 → 65124 [ACK] Seq=61 Ack=378 Win=2160896 Len=0
     76.407667  127.0.0.1   127.0.0.1    TCP           44 9999 → 65124 [FIN, ACK] Seq=61 Ack=378 Win=2160896 Len=0
     76.407715  127.0.0.1   127.0.0.1    TCP           44 65124 → 9999 [ACK] Seq=378 Ack=62 Win=2161152 Len=0
     76.407914  127.0.0.1   127.0.0.1    TCP          104 9998 → 65123 [PSH, ACK] Seq=1 Ack=377 Win=2160896 Len=60
     76.407952  127.0.0.1   127.0.0.1    TCP           44 65123 → 9998 [ACK] Seq=377 Ack=61 Win=2161152 Len=0
     76.408773  127.0.0.1   127.0.0.1    TCP           44 65123 → 9998 [FIN, ACK] Seq=377 Ack=61 Win=2161152 Len=0
     76.408817  127.0.0.1   127.0.0.1    TCP           44 9998 → 65123 [ACK] Seq=61 Ack=378 Win=2160896 Len=0
     76.408833  127.0.0.1   127.0.0.1    TCP           44 9998 → 65123 [FIN, ACK] Seq=61 Ack=378 Win=2160896 Len=0
     76.408908  127.0.0.1   127.0.0.1    TCP           44 65123 → 9998 [ACK] Seq=378 Ack=62 Win=2161152 Len=0
```

Here the process is the same as the first query, because this is a new query, which the response for is not stored in the cache. Therefore, there is a cache "miss" and the connection between the proxy and the server is established just like in the first time.

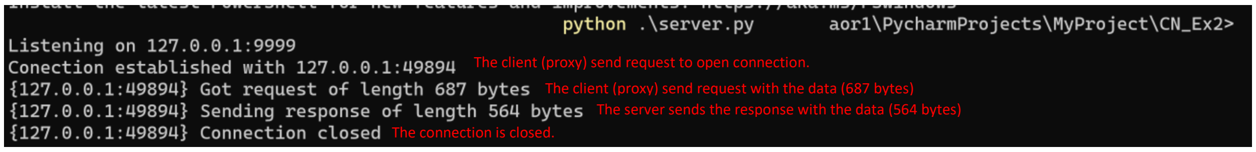
The only difference is in the size (len) of the query data and response data.

3.3.

<u>Screenshots:</u>

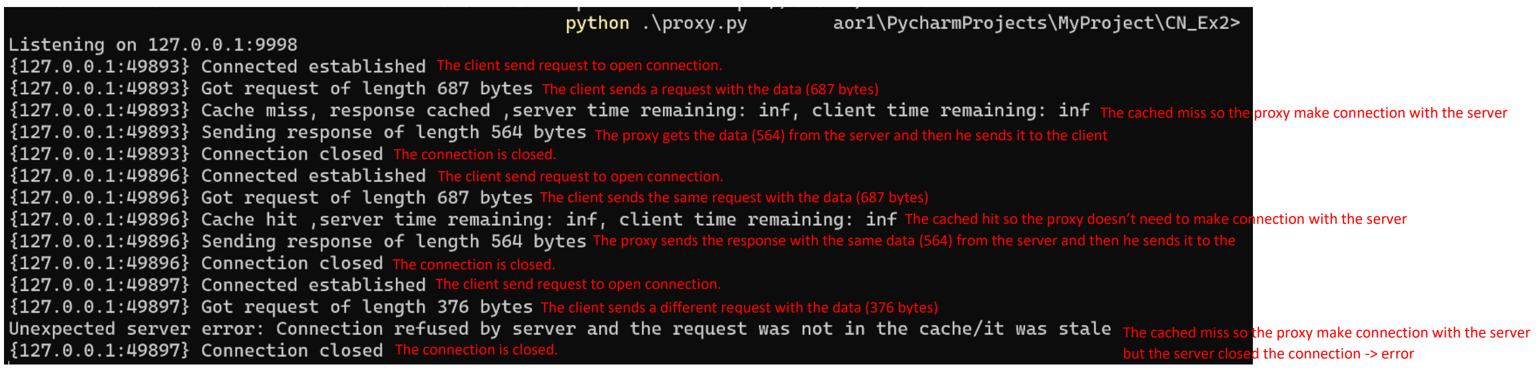
Cmd: Note that our notes are in red. We shout down the server after the first request.

The server:



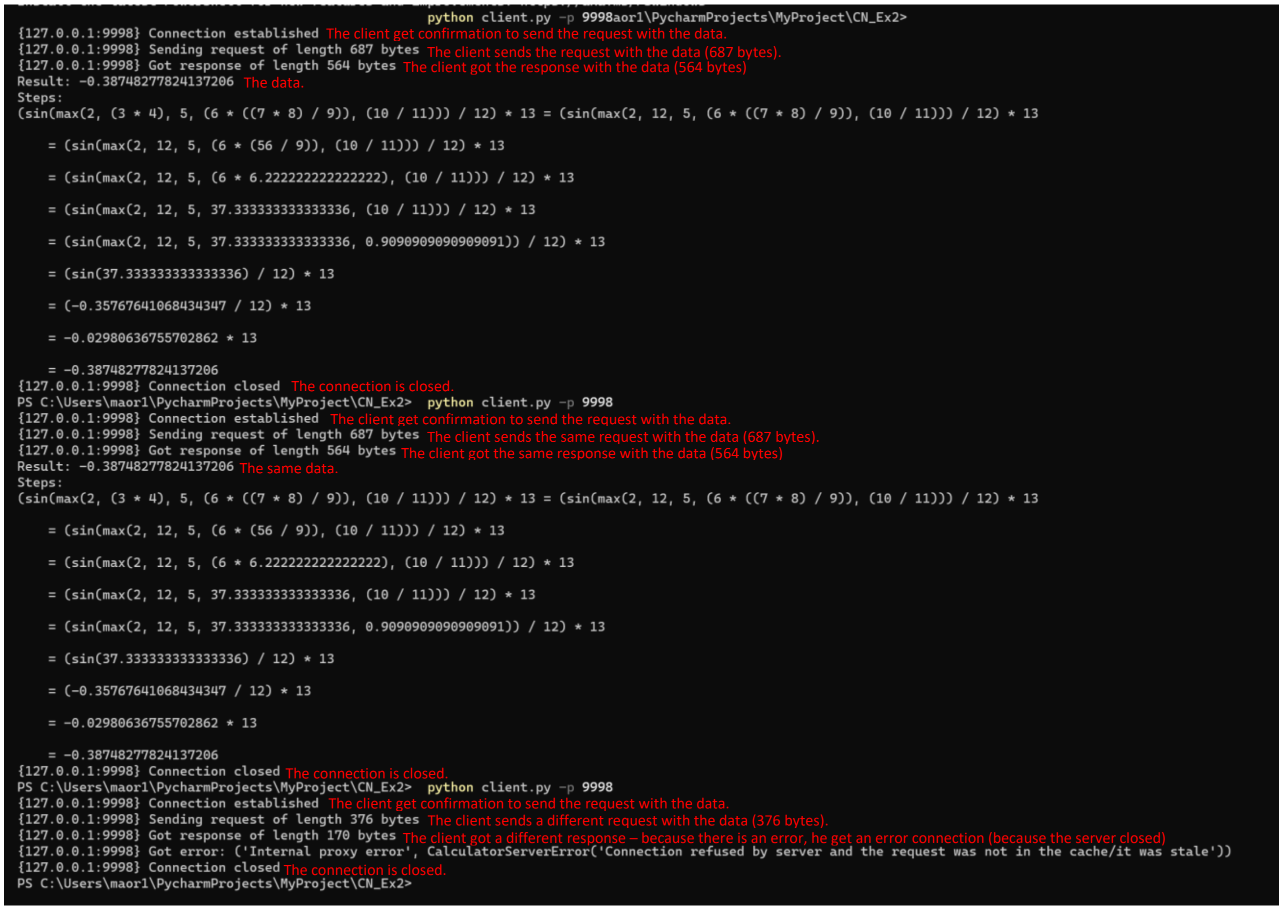
```
                                                      python .\server.py       aor1\PycharmProjects\MyProject\CN_Ex2>
Listening on 127.0.0.1:9999
Conection established with 127.0.0.1:49894  The client (proxy) send request to open connection.
{127.0.0.1:49894} Got request of length 687 bytes  The client (proxy) send request with the data (687 bytes)
{127.0.0.1:49894} Sending response of length 564 bytes  The server sends the response with the data (564 bytes)
{127.0.0.1:49894} Connection closed  The connection is closed.
```

The proxy:



```
                                                      python .\proxy.py        aor1\PycharmProjects\MyProject\CN_Ex2>
Listening on 127.0.0.1:9998
{127.0.0.1:49893} Connected established  The client send request to open connection.
{127.0.0.1:49893} Got request of length 687 bytes  The client sends a request with the data (687 bytes)
{127.0.0.1:49893} Cache miss, response cached ,server time remaining: inf, client time remaining: inf  The cached miss so the proxy make connection with the server
{127.0.0.1:49893} Sending response of length 564 bytes  The proxy gets the data (564) from the server and then he sends it to the client
{127.0.0.1:49893} Connection closed  The connection is closed.
{127.0.0.1:49896} Connected established  The client send request to open connection.
{127.0.0.1:49896} Got request of length 687 bytes  The client sends the same request with the data (687 bytes)
{127.0.0.1:49896} Cache hit ,server time remaining: inf, client time remaining: inf  The cached hit so the proxy doesn't need to make connection with the server
{127.0.0.1:49896} Sending response of length 564 bytes  The proxy sends the response with the same data (564) from the server and then he sends it to the
{127.0.0.1:49896} Connection closed  The connection is closed.
{127.0.0.1:49897} Connected established  The client send request to open connection.
{127.0.0.1:49897} Got request of length 376 bytes  The client sends a different request with the data (376 bytes)
Unexpected server error: Connection refused by server and the request was not in the cache/it was stale  The cached miss so the proxy make connection with the server
{127.0.0.1:49897} Connection closed  The connection is closed.                                                   but the server closed the connection -> error
```

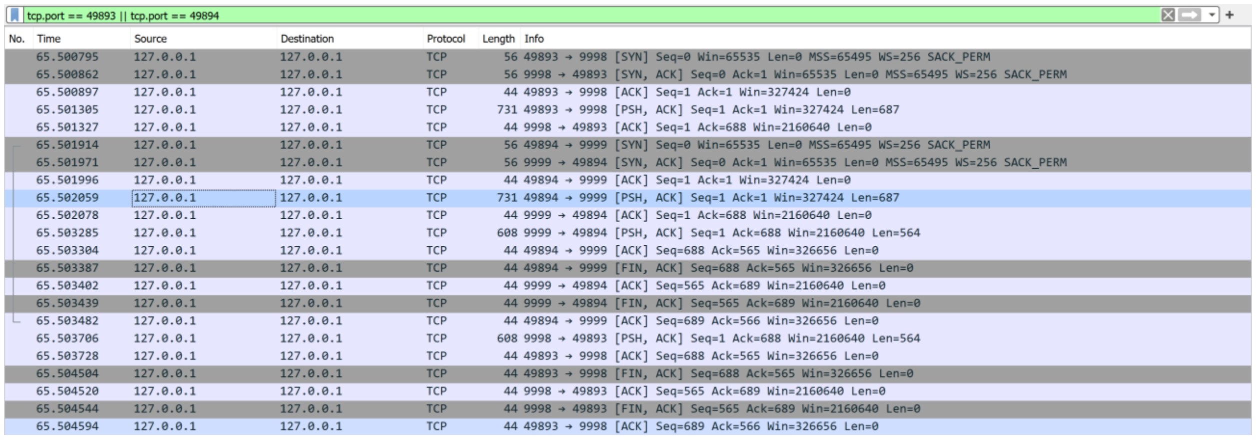
The client: here the client is the client, and the server is the proxy.



```
                                                       python client.py -p 9998aor1\PycharmProjects\MyProject\CN_Ex2>
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 687 bytes The client sends the request with the data (687 bytes).
{127.0.0.1:9998} Got response of length 564 bytes The client got the response with the data (564 bytes)
Result: -0.38748277824137206 The data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.02980636755702862 * 13

    = -0.38748277824137206
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2>  python client.py -p 9998
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 687 bytes The client sends the same request with the data (687 bytes).
{127.0.0.1:9998} Got response of length 564 bytes The client got the same response with the data (564 bytes)
Result: -0.38748277824137206 The same data.
Steps:
(sin(max(2, (3 * 4), 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13 = (sin(max(2, 12, 5, (6 * ((7 * 8) / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * (56 / 9)), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, (6 * 6.222222222222222), (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, (10 / 11))) / 12) * 13

    = (sin(max(2, 12, 5, 37.333333333333336, 0.9090909090909091)) / 12) * 13

    = (sin(37.333333333333336) / 12) * 13

    = (-0.35767641068434347 / 12) * 13

    = -0.02980636755702862 * 13

    = -0.38748277824137206
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2>  python client.py -p 9998
{127.0.0.1:9998} Connection established The client get confirmation to send the request with the data.
{127.0.0.1:9998} Sending request of length 376 bytes The client sends a different request with the data (376 bytes).
{127.0.0.1:9998} Got response of length 170 bytes The client got a different response – because there is an error, he get an error connection (because the server closed)
{127.0.0.1:9998} Got error: ('Internal proxy error', CalculatorServerError('Connection refused by server and the request was not in the cache/it was stale'))
{127.0.0.1:9998} Connection closed The connection is closed.
PS C:\Users\maor1\PycharmProjects\MyProject\CN_Ex2>
```

(1) First request: client ->49893, server(proxy) ->9998

client(proxy) ->49894, server ->9999

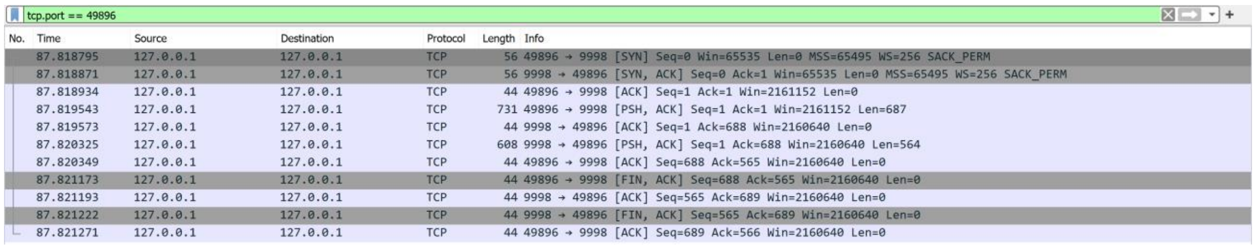


tcp.port == 49893 || tcp.port == 49894

No.	Time	Source	Destination	Protocol	Length	Info
	65.500795	127.0.0.1	127.0.0.1	TCP	56	49893 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	65.500862	127.0.0.1	127.0.0.1	TCP	56	9998 → 49893 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	65.500897	127.0.0.1	127.0.0.1	TCP	44	49893 → 9998 [ACK] Seq=1 Ack=1 Win=327424 Len=0
	65.501305	127.0.0.1	127.0.0.1	TCP	731	49893 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=687
	65.501327	127.0.0.1	127.0.0.1	TCP	44	9998 → 49893 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
	65.501914	127.0.0.1	127.0.0.1	TCP	56	49894 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	65.501971	127.0.0.1	127.0.0.1	TCP	56	9999 → 49894 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	65.501996	127.0.0.1	127.0.0.1	TCP	44	49894 → 9999 [ACK] Seq=1 Ack=1 Win=327424 Len=0
	65.502059	127.0.0.1	127.0.0.1	TCP	731	49894 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=687
	65.502078	127.0.0.1	127.0.0.1	TCP	44	9999 → 49894 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
	65.503285	127.0.0.1	127.0.0.1	TCP	608	9999 → 49894 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
	65.503304	127.0.0.1	127.0.0.1	TCP	44	49894 → 9999 [ACK] Seq=688 Ack=565 Win=326656 Len=0
	65.503387	127.0.0.1	127.0.0.1	TCP	44	49894 → 9999 [FIN, ACK] Seq=688 Ack=565 Win=326656 Len=0
	65.503402	127.0.0.1	127.0.0.1	TCP	44	9999 → 49894 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
	65.503439	127.0.0.1	127.0.0.1	TCP	44	9999 → 49894 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
	65.503482	127.0.0.1	127.0.0.1	TCP	44	49894 → 9999 [ACK] Seq=689 Ack=566 Win=326656 Len=0
	65.503706	127.0.0.1	127.0.0.1	TCP	608	9998 → 49893 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
	65.503728	127.0.0.1	127.0.0.1	TCP	44	49893 → 9998 [ACK] Seq=688 Ack=565 Win=326656 Len=0
	65.504504	127.0.0.1	127.0.0.1	TCP	44	49893 → 9998 [FIN, ACK] Seq=688 Ack=565 Win=326656 Len=0
	65.504520	127.0.0.1	127.0.0.1	TCP	44	9998 → 49893 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
	65.504544	127.0.0.1	127.0.0.1	TCP	44	9998 → 49893 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
	65.504594	127.0.0.1	127.0.0.1	TCP	44	49893 → 9998 [ACK] Seq=689 Ack=566 Win=326656 Len=0

Here the explanation is identical to the one explaining the process in question 3.2.1

(2) First request: client ->49896, server(proxy) ->9998



tcp.port == 49896

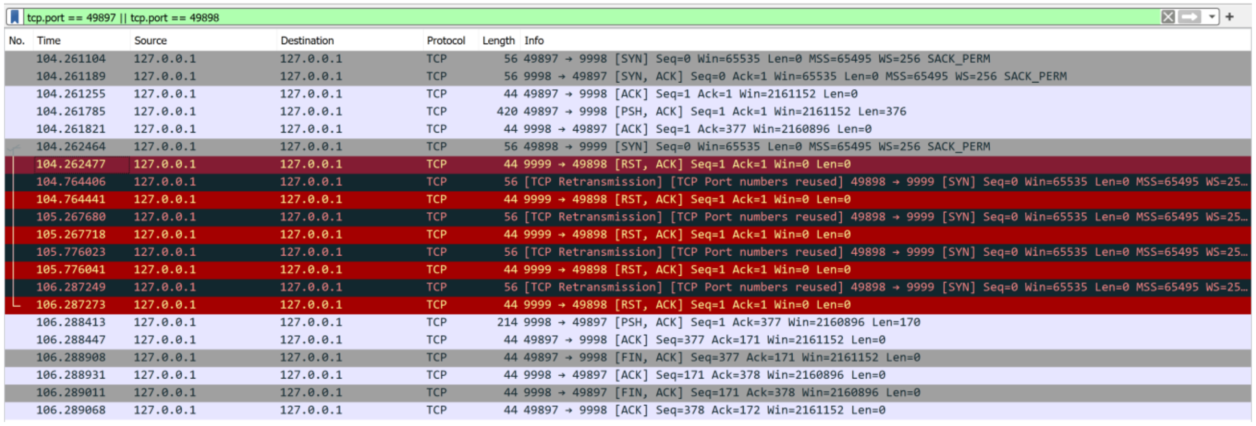
No.	Time	Source	Destination	Protocol	Length	Info
	87.818795	127.0.0.1	127.0.0.1	TCP	56	49896 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	87.818871	127.0.0.1	127.0.0.1	TCP	56	9998 → 49896 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
	87.818934	127.0.0.1	127.0.0.1	TCP	44	49896 → 9998 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
	87.819543	127.0.0.1	127.0.0.1	TCP	731	49896 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=687
	87.819573	127.0.0.1	127.0.0.1	TCP	44	9998 → 49896 [ACK] Seq=1 Ack=688 Win=2160640 Len=0
	87.820325	127.0.0.1	127.0.0.1	TCP	608	9998 → 49896 [PSH, ACK] Seq=1 Ack=688 Win=2160640 Len=564
	87.820349	127.0.0.1	127.0.0.1	TCP	44	49896 → 9998 [ACK] Seq=688 Ack=565 Win=2160640 Len=0
	87.821173	127.0.0.1	127.0.0.1	TCP	44	49896 → 9998 [FIN, ACK] Seq=688 Ack=565 Win=2160640 Len=0
	87.821193	127.0.0.1	127.0.0.1	TCP	44	9998 → 49896 [ACK] Seq=565 Ack=689 Win=2160640 Len=0
	87.821222	127.0.0.1	127.0.0.1	TCP	44	9998 → 49896 [FIN, ACK] Seq=565 Ack=689 Win=2160640 Len=0
	87.821271	127.0.0.1	127.0.0.1	TCP	44	49896 → 9998 [ACK] Seq=689 Ack=566 Win=2160640 Len=0

Here the explanation is identical to the one explaining the process in question 3.2.2

Although, in this attempt the server is already closed. This does not affect the process in any way, since this is the same query as before, and therefore there is caches "hit" and no connection with the server is needed, just as explained in question 3.2.2.

(3) First request: client ->49897, server(proxy) ->9998

client(proxy) ->49898, server ->9999



```
tcp.port == 49897 || tcp.port == 49898

No. Time        Source      Destination  Protocol Length Info
    104.261104  127.0.0.1   127.0.0.1    TCP       56 49897 → 9998 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    104.261189  127.0.0.1   127.0.0.1    TCP       56 9998 → 49897 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    104.261255  127.0.0.1   127.0.0.1    TCP       44 49897 → 9998 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
    104.261785  127.0.0.1   127.0.0.1    TCP      420 49897 → 9998 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=376
    104.261821  127.0.0.1   127.0.0.1    TCP       44 9998 → 49897 [ACK] Seq=1 Ack=377 Win=2160896 Len=0
    104.262464  127.0.0.1   127.0.0.1    TCP       56 49898 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
    104.262477  127.0.0.1   127.0.0.1    TCP       44 9999 → 49898 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    104.764406  127.0.0.1   127.0.0.1    TCP       56 [TCP Retransmission] [TCP Port numbers reused] 49898 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25…
    104.764441  127.0.0.1   127.0.0.1    TCP       44 9999 → 49898 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    105.267680  127.0.0.1   127.0.0.1    TCP       56 [TCP Retransmission] [TCP Port numbers reused] 49898 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25…
    105.267718  127.0.0.1   127.0.0.1    TCP       44 9999 → 49898 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    105.776023  127.0.0.1   127.0.0.1    TCP       56 [TCP Retransmission] [TCP Port numbers reused] 49898 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25…
    105.776041  127.0.0.1   127.0.0.1    TCP       44 9999 → 49898 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    106.287249  127.0.0.1   127.0.0.1    TCP       56 [TCP Retransmission] [TCP Port numbers reused] 49898 → 9999 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25…
    106.287273  127.0.0.1   127.0.0.1    TCP       44 9999 → 49898 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    106.288413  127.0.0.1   127.0.0.1    TCP      214 9998 → 49897 [PSH, ACK] Seq=1 Ack=377 Win=2160896 Len=170
    106.288447  127.0.0.1   127.0.0.1    TCP       44 49897 → 9998 [ACK] Seq=377 Ack=171 Win=2161152 Len=0
    106.288908  127.0.0.1   127.0.0.1    TCP       44 49897 → 9998 [FIN, ACK] Seq=377 Ack=171 Win=2161152 Len=0
    106.288931  127.0.0.1   127.0.0.1    TCP       44 9998 → 49897 [ACK] Seq=171 Ack=378 Win=2160896 Len=0
    106.289011  127.0.0.1   127.0.0.1    TCP       44 9998 → 49897 [FIN, ACK] Seq=171 Ack=378 Win=2160896 Len=0
    106.289068  127.0.0.1   127.0.0.1    TCP       44 49897 → 9998 [ACK] Seq=378 Ack=172 Win=2161152 Len=0
```

Here, we send a different query to the proxy, and therefore there is a cache "miss". For this reason, the proxy attempts to establish a connection with the server. But the server is closed, so no such connection is available.

We can see in all the packages colored black, that the proxy reattempts several times to establish a connection with the server, without success.
The red packages show us that no ACK is received from the server to the proxy.
finally the proxy sends a response to the client stating that there no connection could be established with the server and proceeds to end the connection with the client.