Zeynepnur ŞAHİNEL
2305399 14.05.2022

EE444 Introduction to Computer Networks

HW1-Socket Programming

In this homework, a simple proxy server system consisting of 3 nodes is implemented, that are Client, Proxy, and Server. Using Python, socket programming of the client and the proxy_server is complemented. Hereby, critical steps for both proxy_server and the client are explained respectively. However, I only could construct the server and the client. *Therefore it is not a simple proxy server, it is a simple server-client system*. Thus, in the below explanations, it should be noted that the parts mentioned as proxy_server should be thought as simple server.

Inside the proxy server:
Initially a socket, later providing communication between the client, is created. Here using socket.gethostbyname() function local IPv4 address is obtained for the server IP and a random port is entered for the address (IP, Port). After constructing the address, created proxy_server socket is binded to this address. In the code, two functions are implemented being start() and handle_client(). Inside the start(), listening occurs continuously. While listening continues it accepts the connection from the client with accept(). Through this function, a socket object conn and a new address are created. Connection is established after accept. Moreover, threads are created to deal handle_client() function simultaneously. In the handle_client() coming messages from the client are processed according to their operations. This function can be considered as a connected state (Fig 1. in the HW1 manual)

Inside the client:
Similar to the proxy server, a socket is created and then connected to the address the same as the address of the proxy_server. After connecting, it sends continuously messages in the determined format of the related operations. These messages are sent through a command window.

There are four operations GET, PUT, CLR and ADD. Below, all of these operations are going to be investigated.
GET:
Client message format for GET: OP=GET;IND=Ind1,Ind2,Ind3
Proxy message format for GET: OP=GET;IND=Ind1,Ind2,Ind3,....;Data1,Data2,Data3,....

PUT:
Client message format for PUT: OP=PUT;IND=Ind1,Ind2,Ind3,.....;Datax,Datay,Dataz,.....
Proxy message format for PUT: “PUT operation is completed. Server prints the new table.”
Here, since table is changed, in the proxy_server terminal, new table can be seen.

Zeynepnur ŞAHİNEL
2305399 14.05.2022

ADD:
Client message format for ADD: OP=PUT;IND=Ind1,Ind2,Ind3,.....
Proxy message format for ADD: OP=GET;IND=Ind1,Ind2,Ind3,....;Datax

CLR:
Client message format for CLR: OP=CLR
Proxy message format for CLR: “CLR operation is completed. Server prints the new table.”

To close the socket, the disconnect message is used (!DISCONNECT) Again, this message is given through a command window.

From Fig.1, Fig. 2 and Fig3, OP results can be seen. Fig.4 shows the operations in shown in Fig1, Fig and Fig.3 and Table 1 shows the initial stored table inside the server.

Table 1. Stored Table Index and Data

Index	**Data (Integer)**
0	10
1	11
2	12
3	13
4	14
5	15
6	16
7	17
8	18
9	19

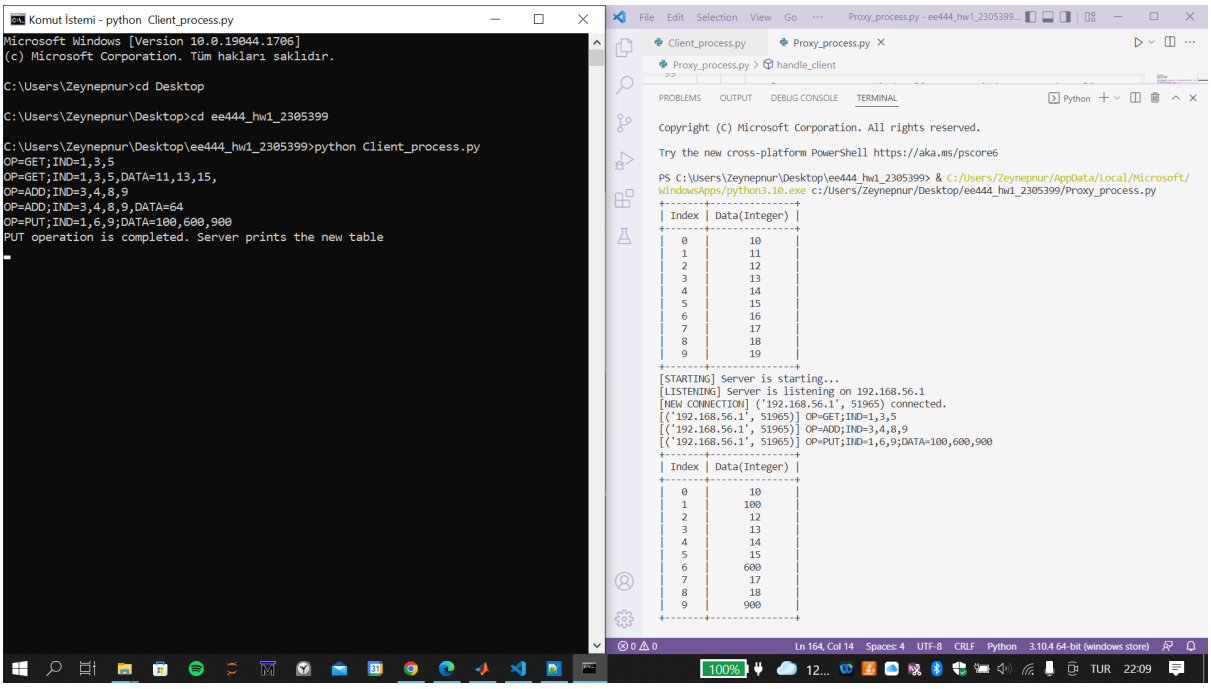

Figure 1. GET, PUT and ADD Operation Results on Client and Proxy_Server Terminals

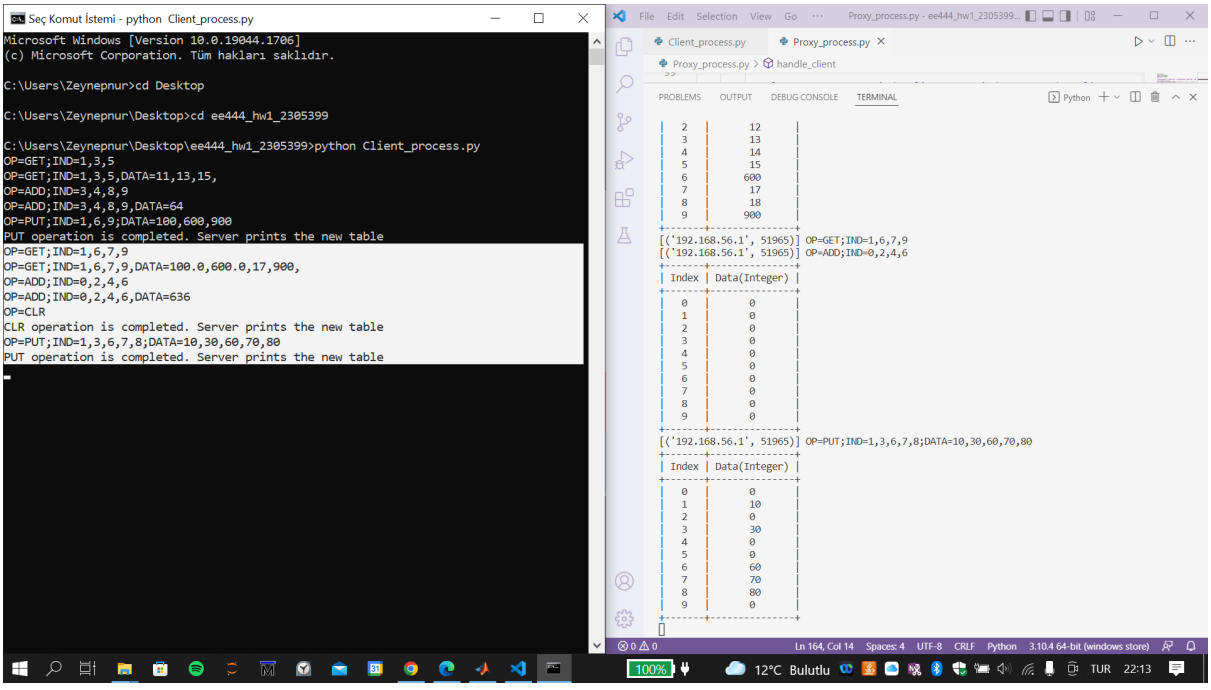

Figure 2. GET, PUT, ADD, CLR Operation Results on Client and Proxy_Server Terminals

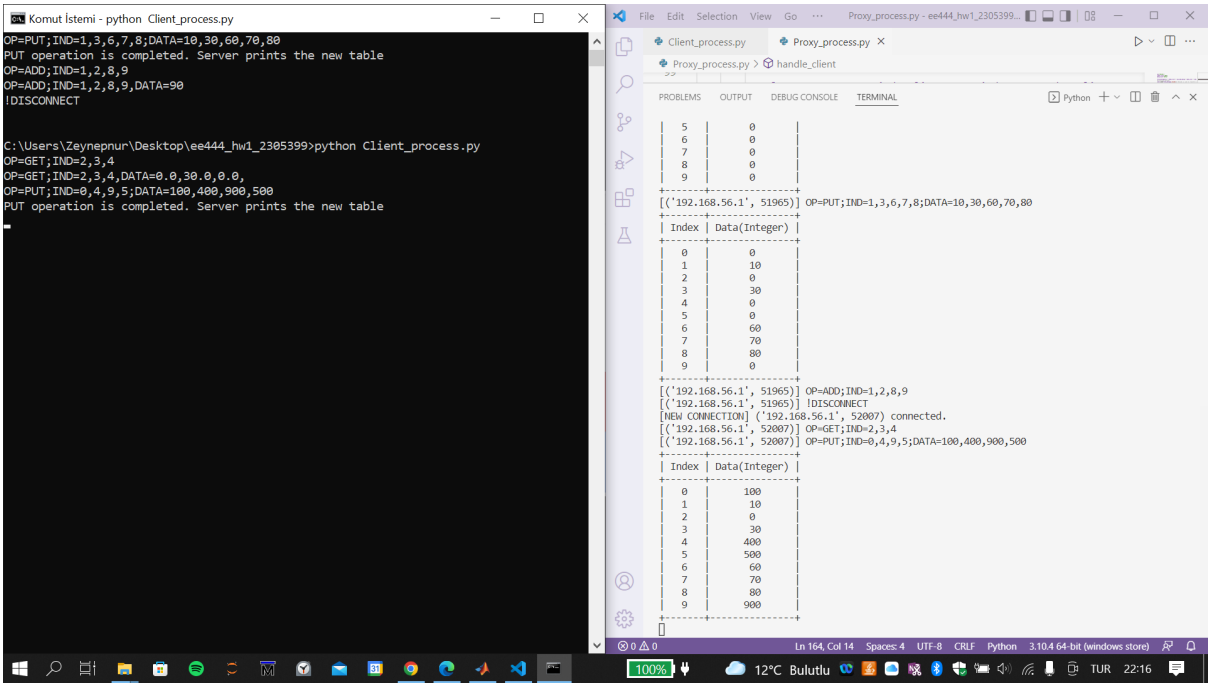

Figure 3. Disconnection of the Socket and a New Connection Creation

Zeynepnur ŞAHİNEL
2305399 14.05.2022

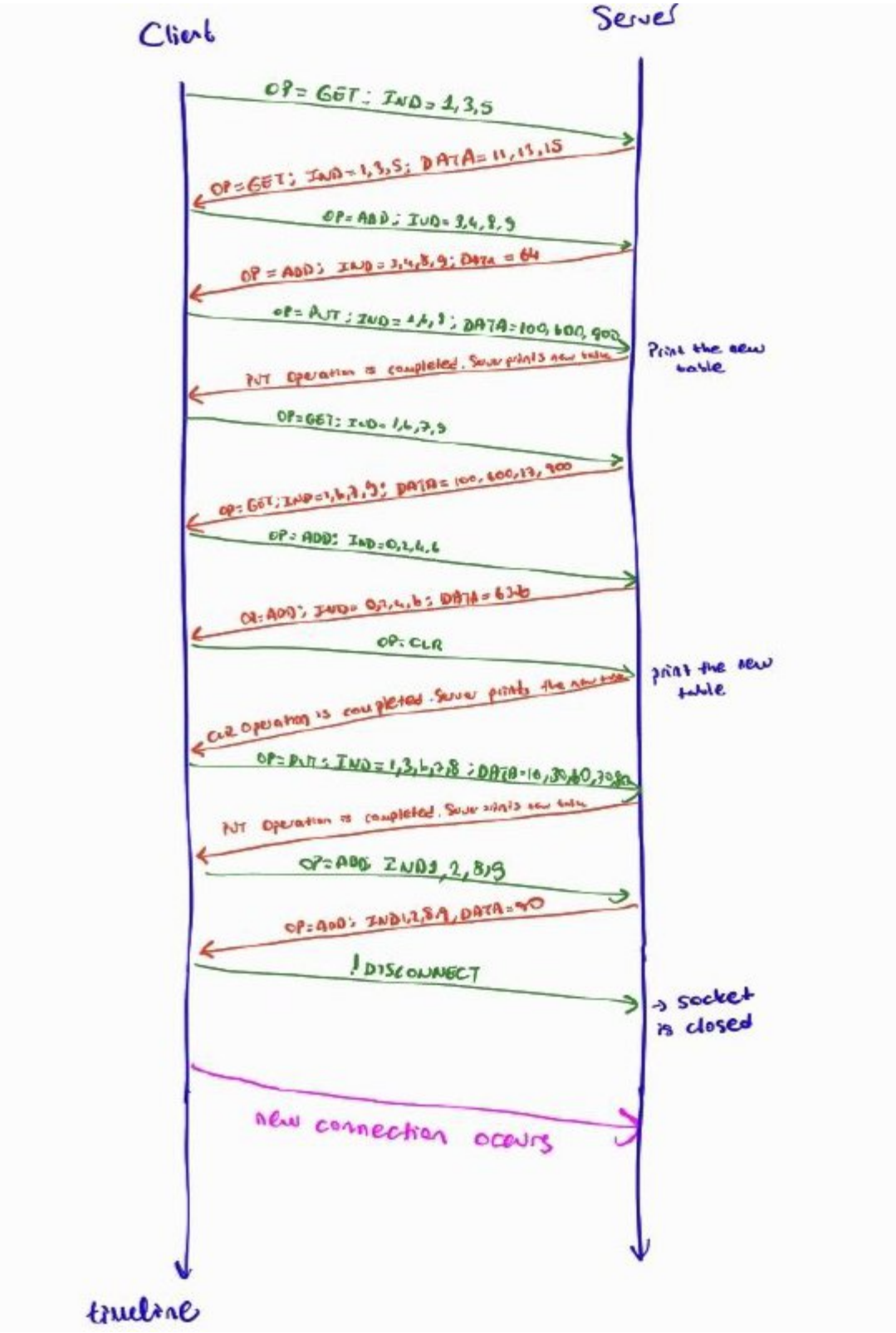

Figure 4. Operation timeline and client-proxy messages

Reference:

https://www.youtube.com/watch?v=3QiPPX-KeSc&t=2449s&ab_channel=TechWithTim