

## 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.

**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

The image shows two terminal windows. The left window is the 'Client\_process.py' terminal, and the right window is the 'Proxy\_process.py' terminal. Both windows show the execution of a Python script that performs GET, PUT, and ADD operations on a server. The output of the script is displayed in the terminal windows, showing the results of the operations and the state of the server's data table.

**Client\_process.py Output:**

```
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\Zeynepnur>cd Desktop
C:\Users\Zeynepnur\Desktop>cd ee444_hw1_2305399
C:\Users\Zeynepnur\Desktop\ee444_hw1_2305399>python Client_process.py
OP=GET;IND=1,3,5
OP=GET;IND=1,3,5;DATA=11,13,15,
OP=ADD;IND=3,4,8,9
OP=ADD;IND=3,4,8,9;DATA=64
OP=PUT;IND=1,6,9;DATA=100,600,900
PUT operation is completed. Server prints the new table
```

**Proxy\_process.py Output:**

```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Zeynepnur\Desktop\ee444_hw1_2305399> & C:\Users\Zeynepnur\AppData\Local\Microsoft\WindowsApps\python3.10.exe c:\Users\Zeynepnur\Desktop\ee444_hw1_2305399\Proxy_process.py

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

[STARTING] Server is starting...
[LISTENING] Server is listening on 192.168.56.1
[NEW CONNECTION] ('192.168.56.1', 51965) connected.
[('192.168.56.1', 51965)] OP=GET;IND=1,3,5
[('192.168.56.1', 51965)] OP=ADD;IND=3,4,8,9
[('192.168.56.1', 51965)] OP=PUT;IND=1,6,9;DATA=100,600,900

Index | Data(Integer) |
-----|-----|
0 | 10 |
1 | 100 |
2 | 12 |
3 | 13 |
4 | 14 |
5 | 15 |
6 | 600 |
7 | 17 |
8 | 18 |
9 | 900 |
```

Figure 1. GET, PUT and ADD Operation Results on Client and Proxy\_Server Terminals

The image shows two terminal windows. The left window is the 'Client\_process.py' terminal, and the right window is the 'Proxy\_process.py' terminal. Both windows show the execution of a Python script that performs GET, PUT, ADD, and CLR operations on a server. The output of the script is displayed in the terminal windows, showing the results of the operations and the state of the server's data table.

**Client\_process.py Output:**

```
Seç Komut İstemi - python Client_process.py
Microsoft Windows [Version 10.0.19044.1706]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\Zeynepnur>cd Desktop
C:\Users\Zeynepnur\Desktop>cd ee444_hw1_2305399
C:\Users\Zeynepnur\Desktop\ee444_hw1_2305399>python Client_process.py
OP=GET;IND=1,3,5
OP=GET;IND=1,3,5;DATA=11,13,15,
OP=ADD;IND=3,4,8,9
OP=ADD;IND=3,4,8,9;DATA=64
OP=PUT;IND=1,6,9;DATA=100,600,900
PUT operation is completed. Server prints the new table
OP=GET;IND=1,6,7,9
OP=GET;IND=1,6,7,9;DATA=100,0,600,0,17,900,
OP=ADD;IND=0,2,4,6
OP=ADD;IND=0,2,4,6;DATA=636
OP=CLR
CLR operation is completed. Server prints the new table
OP=PUT;IND=1,3,6,7,8;DATA=10,30,60,70,80
PUT operation is completed. Server prints the new table
```

**Proxy\_process.py Output:**

```
Index | Data(Integer) |
-----|-----|
2 | 12 |
3 | 13 |
4 | 14 |
5 | 15 |
6 | 600 |
7 | 17 |
8 | 18 |
9 | 900 |

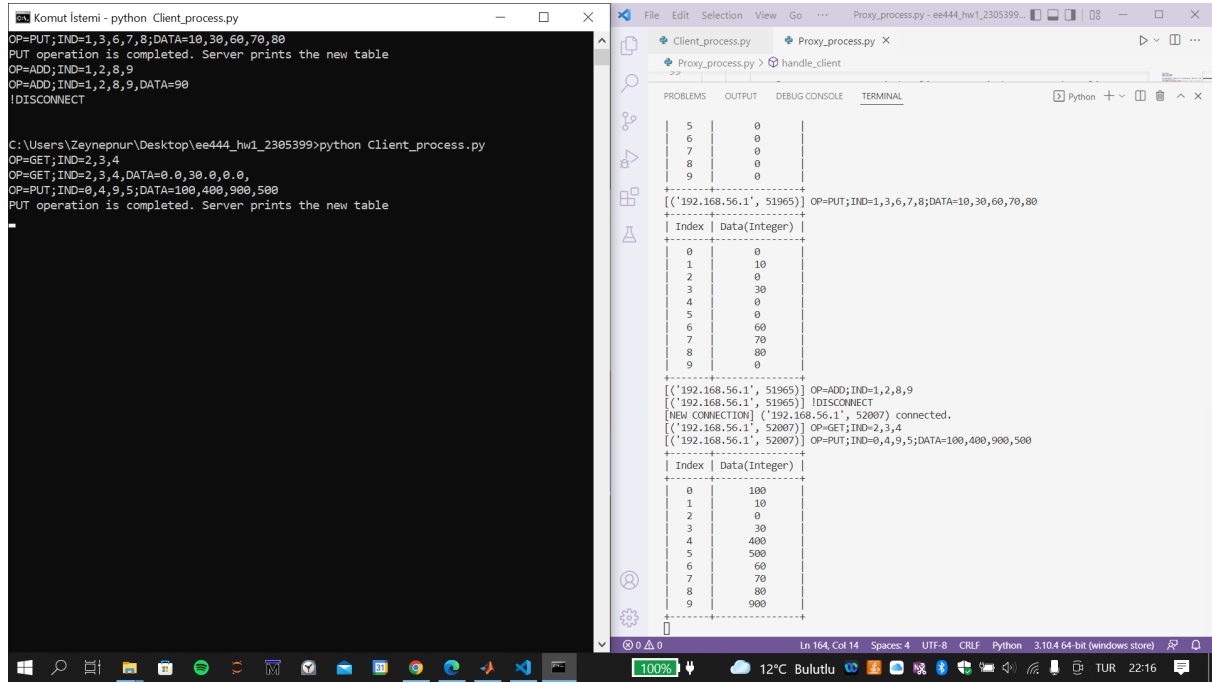
[('192.168.56.1', 51965)] OP=GET;IND=1,6,7,9
[('192.168.56.1', 51965)] OP=ADD;IND=0,2,4,6

Index | Data(Integer) |
-----|-----|
0 | 0 |
1 | 0 |
2 | 0 |
3 | 0 |
4 | 0 |
5 | 0 |
6 | 0 |
7 | 0 |
8 | 0 |
9 | 0 |

[('192.168.56.1', 51965)] OP=PUT;IND=1,3,6,7,8;DATA=10,30,60,70,80

Index | Data(Integer) |
-----|-----|
0 | 0 |
1 | 10 |
2 | 0 |
3 | 30 |
4 | 0 |
5 | 0 |
6 | 60 |
7 | 70 |
8 | 80 |
9 | 0 |
```

Figure 2. GET, PUT, ADD, CLR Operation Results on Client and Proxy\_Server Terminals



```
Komut İstemi - python Client_process.py
OP=PUT;IND=1,3,6,7,8;DATA=10,30,60,70,80
PUT operation is completed. Server prints the new table
OP=ADD;IND=1,2,8,9
OP=ADD;IND=1,2,8,9,DATA=90
!DISCONNECT

C:\Users\Zeynepnur\Desktop\ee444_hw1_2305399>python Client_process.py
OP=GET;IND=2,3,4
OP=GET;IND=2,3,4,DATA=0,0,30,0,0,0,0,
OP=PUT;IND=0,4,0,5;DATA=100,400,900,500
PUT operation is completed. Server prints the new table
```

```
Client_process.py  Proxy_process.py
Proxy_process.py > handle_client

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

5 | 0
6 | 0
7 | 0
8 | 0
9 | 0

[('192.168.56.1', 51965)] OP=PUT;IND=1,3,6,7,8;DATA=10,30,60,70,80

Index | Data(Integer)
-----|-----
0 | 0
1 | 10
2 | 0
3 | 30
4 | 0
5 | 0
6 | 60
7 | 70
8 | 80
9 | 0

[('192.168.56.1', 51965)] OP=ADD;IND=1,2,8,9
[('192.168.56.1', 51965)] !DISCONNECT
[NEW CONNECTION] ('192.168.56.1', 52007) connected.
[('192.168.56.1', 52007)] OP=GET;IND=2,3,4
[('192.168.56.1', 52007)] OP=PUT;IND=0,4,9,5;DATA=100,400,900,500

Index | Data(Integer)
-----|-----
0 | 100
1 | 10
2 | 0
3 | 30
4 | 400
5 | 500
6 | 60
7 | 70
8 | 80
9 | 900
```

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

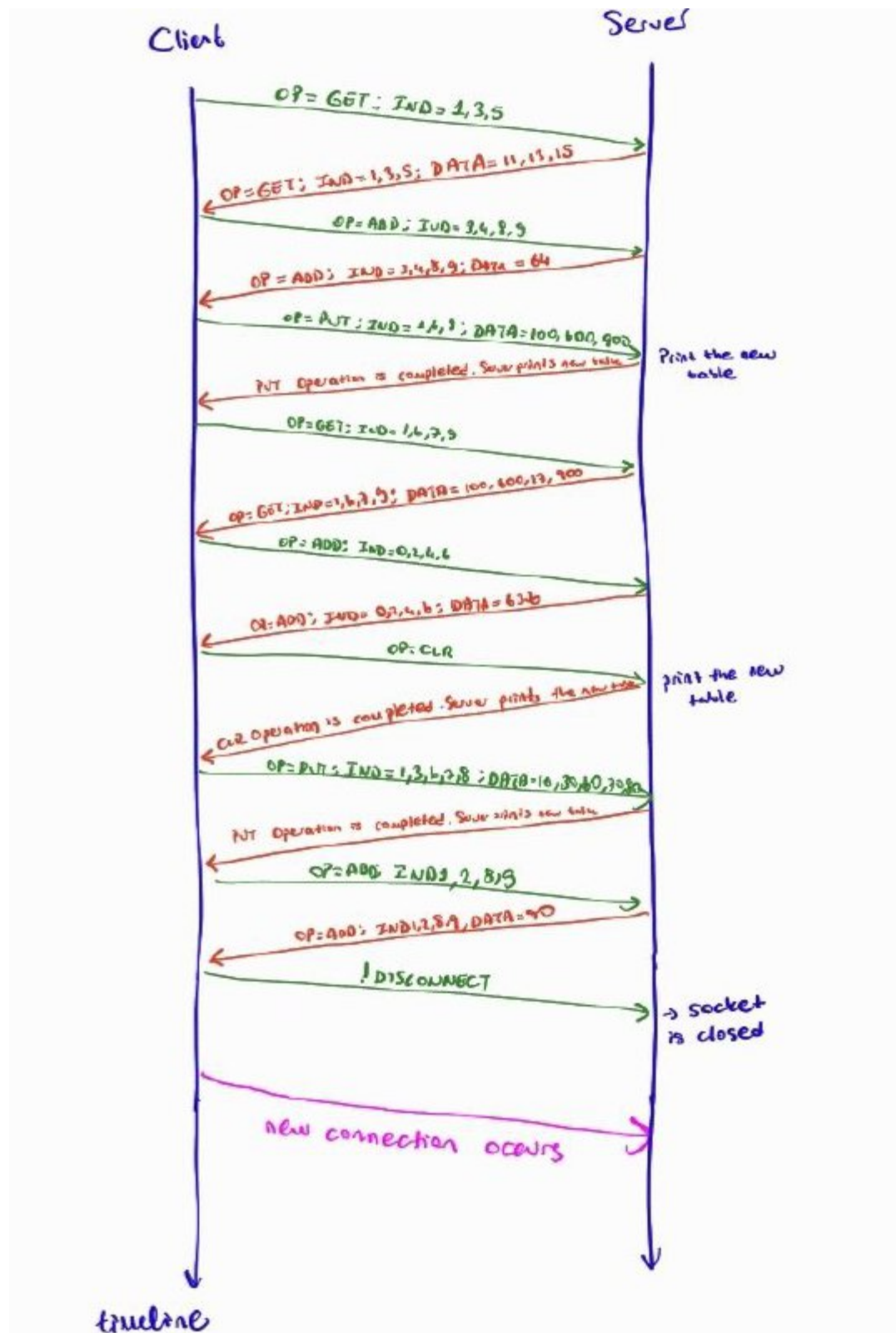


Figure 4. Operation timeline and client-proxy messages

#### Reference:

[https://www.youtube.com/watch?v=3QiPPX-KeSc&t=2449s&ab\\_channel=TechWithTim](https://www.youtube.com/watch?v=3QiPPX-KeSc&t=2449s&ab_channel=TechWithTim)