CSE 328
Computer Networks

Project 1

Socket Programming

18/3/2023

Submitted by

Youssef Ashraf El-Harty – CSE 2 – 120200090

Ahmed Hagag Abulwafa – CSE 2 – 120200101

to

Dr. Ahmed Fares

TCP client and server sockets:

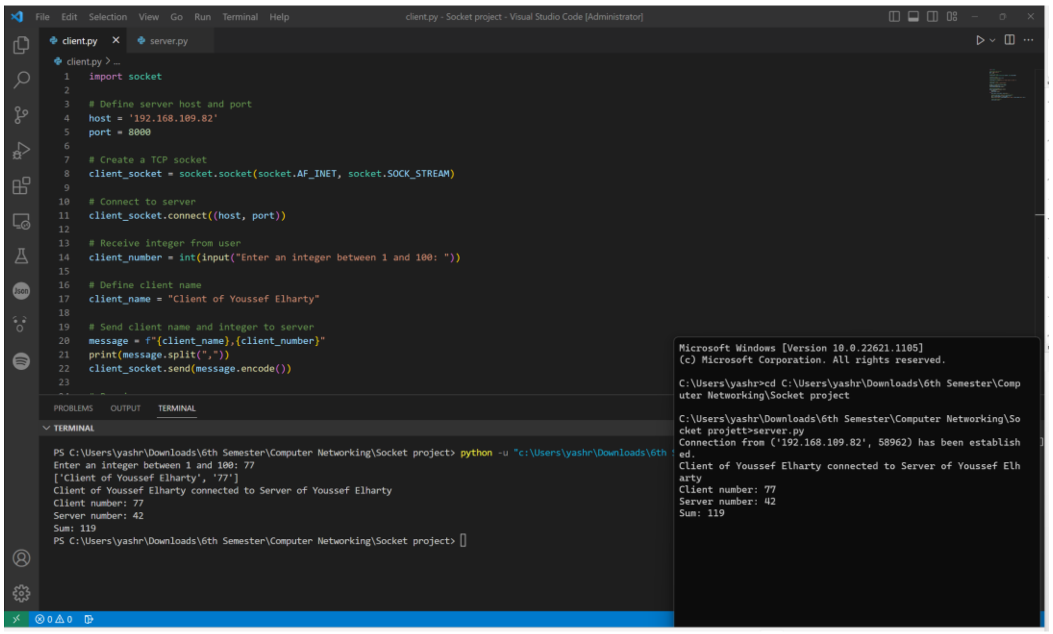
In this part of the project, we created two different pieces of codes, a client and a server socket. Each file was run on a different computer to simulate a client and a server scenario.

The client code uses TCP connection, it sends a request to the TCP socket to initiate a connection, then the server code listens for a connection from the client. The user in the client end then enters a number from 1 to 100. If the number is more than 100 or less than 1, the server prints an error message and ends the connection.

The client socket then sends a formatted message to the server. The server receives the message and print the client name, the server name, client number, and the server number then it computes the sum and prints it.

The server sends a message to the client containing the name of the server, and its number. Then the client computes the sum of the two numbers and prints along with the other data.

Here is a screenshot showing the two codes running.


File Edit Selection View Go Run Terminal Help
client.py - Socket project - Visual Studio Code [Administrator]
client.py
server.py
client.py > ...
PROBLEMS
OUTPUT
TERMINAL

```python
import socket

# Define server host and port
host = '192.168.109.82'
port = 8000

# Create a TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to server
client_socket.connect((host, port))

# Receive integer from user
client_number = int(input("Enter an integer between 1 and 100: "))

# Define client name
client_name = "Client of Youssef Elharty"

# Send client name and integer to server
message = f"{client_name},{client_number}"
print(message.split(","))
client_socket.send(message.encode())

```

```
PS C:\Users\yashr\Downloads\6th Semester\Computer Networking\Socket project> python -u "c:\Users\yashr\Downloads\6th
Enter an integer between 1 and 100: 77
['Client of Youssef Elharty', '77']
Client of Youssef Elharty connected to Server of Youssef Elharty
Client number: 77
Server number: 42
Sum: 119
PS C:\Users\yashr\Downloads\6th Semester\Computer Networking\Socket project> 
```

```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yashr>cd C:\Users\yashr\Downloads\6th Semester\Comp
uter Networking\Socket project

C:\Users\yashr\Downloads\6th Semester\Computer Networking\So
cket projett>server.py
Connection from ('192.168.109.82', 58962) has been establish
ed.
Client of Youssef Elharty connected to Server of Youssef Elh
arty
Client number: 77
Server number: 42
Sum: 119
```

Threading part (Extra)

Server File Code:

This Python file is a server program that listens for incoming connections on a specified IP address and port number and handles client requests.

The socket and threading modules are imported at the beginning of the file.

The count variable is initialized to 0.

The handle_client function takes four parameters: conn (the connection object), addr (the address of the client), server_name (a string representing the name of the server), and server_number (an integer representing a number associated with the server).

Within the function, the incoming client request is received and decoded. If the data is empty, the function breaks out of the loop. Otherwise, the client_name and client_number are extracted from the data string. If the client_number is greater than 100 or less than 0, the function breaks out of the loop.

Next, the function prints a message indicating the client has connected to the server, and prints the client's number. It then calculates the sum of the client_number and server_number, and prints this sum. The modified message is then encoded and sent back to the client.

The start_server function initializes the server by setting the IP address, port number, server_name, and server_number values. A socket object is then created, bound to the specified IP address and port number, and set to listen for incoming connections. The function then enters a loop that continuously accepts incoming client connections, creating a new thread to handle each connection.

Finally, the if __name__ == "__main__": block executes the start_server function if the script is run directly (rather than imported as a module).

Client File Code:

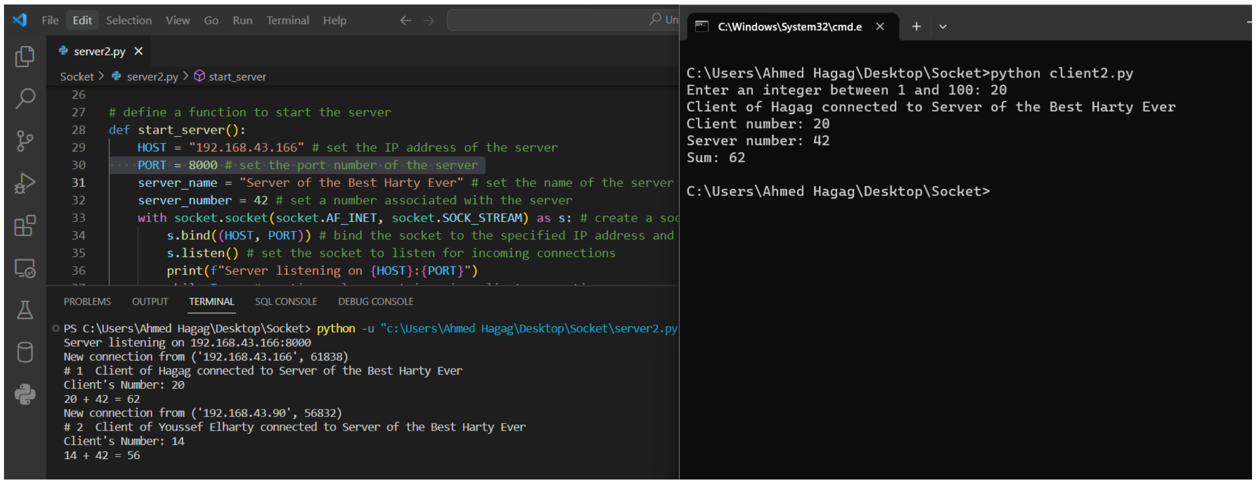
This Python script defines a function called start_client that establishes a socket connection to a remote server and sends data to the server.

The script starts by importing the socket module, which provides low-level network communication. The start_client function sets the IP address and port number of the server to connect to, creates a socket object, and connects to the server using the connect method.

The function then enters an infinite loop that prompts the user to enter an integer between 1 and 100. The integer and a client name are combined into a message and sent to the server using the send method. The client then receives data from the server using the recv method, which is decoded from bytes to a string using the decode method.

The received data contains the server name and number, which are extracted from the message using the split method. The client then calculates the sum of the server and client numbers and prints the client number, server number, and their sum. Finally, the client socket connection is closed using the close method.

If the script is run directly, the start_client function is called, initiating the process of connecting to the server and sending and receiving data.


File Edit Selection View Go Run Terminal Help
server2.py
Socket > server2.py > start_server
PROBLEMS OUTPUT TERMINAL SQL CONSOLE DEBUG CONSOLE
C:\Windows\System32\cmd.e

```python

# define a function to start the server
def start_server():
    HOST = "192.168.43.166" # set the IP address of the server
    PORT = 8000 # set the port number of the server
    server_name = "Server of the Best Harty Ever" # set the name of the server
    server_number = 42 # set a number associated with the server
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s: # create a soc
        s.bind((HOST, PORT)) # bind the socket to the specified IP address and
        s.listen() # set the socket to listen for incoming connections
        print(f"Server listening on {HOST}:{PORT}")
```

```
PS C:\Users\Ahmed Hagag\Desktop\Socket> python -u "c:\Users\Ahmed Hagag\Desktop\Socket\server2.py
Server listening on 192.168.43.166:8000
New connection from ('192.168.43.166', 61838)
# 1  Client of Hagag connected to Server of the Best Harty Ever
Client's Number: 20
20 + 42 = 62
New connection from ('192.168.43.90', 56832)
# 2  Client of Youssef Elharty connected to Server of the Best Harty Ever
Client's Number: 14
14 + 42 = 56
```

```
C:\Users\Ahmed Hagag\Desktop\Socket>python client2.py
Enter an integer between 1 and 100: 20
Client of Hagag connected to Server of the Best Harty Ever
Client number: 20
Server number: 42
Sum: 62

C:\Users\Ahmed Hagag\Desktop\Socket>
```

This screenshot shows two clients with different IP addresses and ports.