Group 16

CS420(G) Computer Networks
Python programming Assignment part 2 of 2

By Sushma Gajulapalli, Sathvika Kumbham, Aishwarya Khatpe, Meghana Kandala

Communication Between Server and Client Using Python Sockets

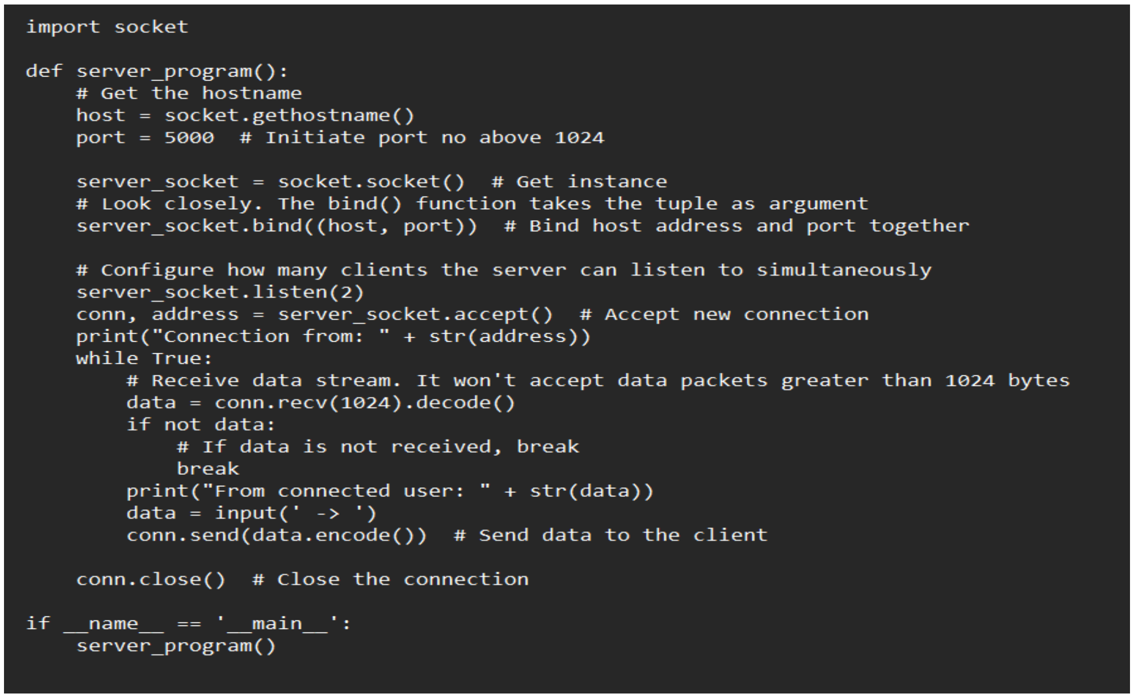
Video explanation: https://youtu.be/qkiegmJdGqE(https://youtu.be/qkiegmJdGqE)

1. Introduction

In network programming, communication between a server and a client is a fundamental concept. Sockets are the primary means for establishing this communication, allowing data exchange between devices over a network. This report explores a simple implementation of server-client communication using Python's socket module.

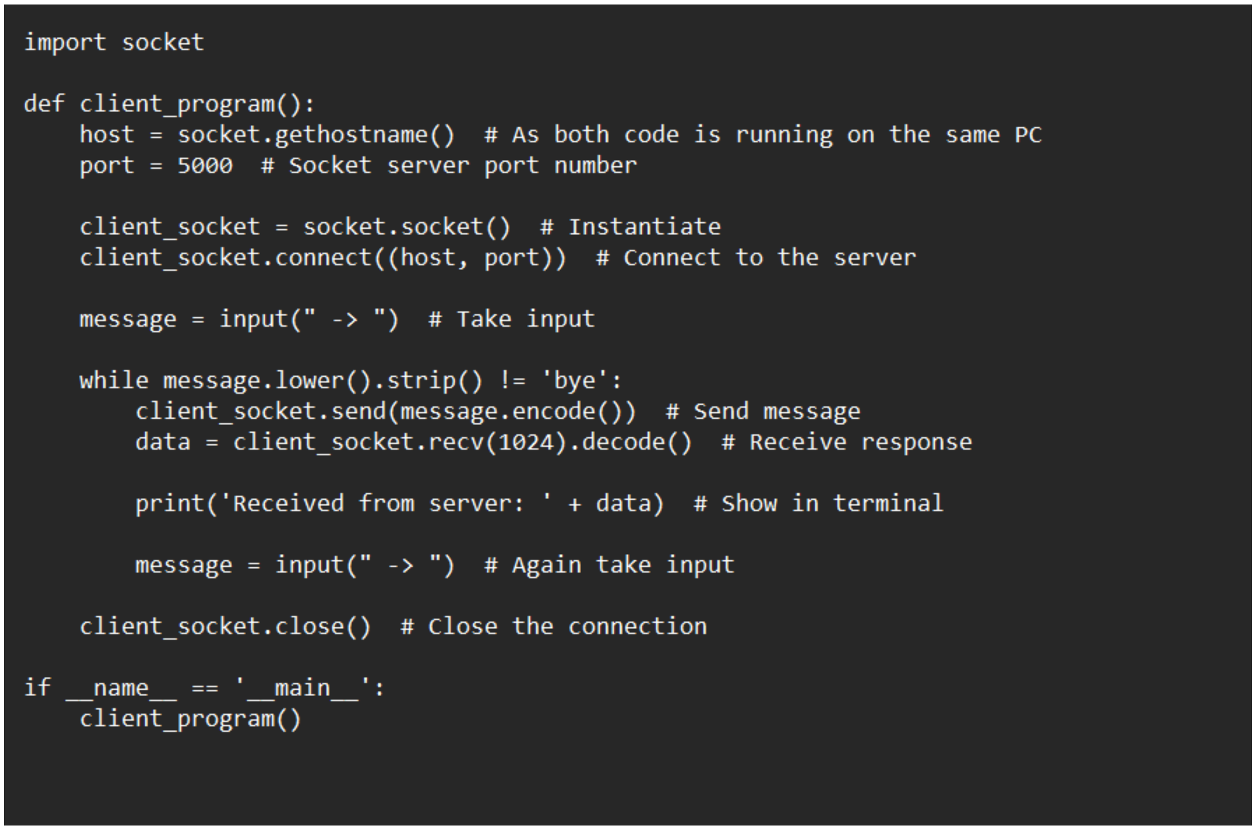
2. Code Implementation

2.1 Server Side (server.py)



```python
import socket

def server_program():
    # Get the hostname
    host = socket.gethostname()
    port = 5000  # Initiate port no above 1024

    server_socket = socket.socket()  # Get instance
    # Look closely. The bind() function takes the tuple as argument
    server_socket.bind((host, port))  # Bind host address and port together

    # Configure how many clients the server can listen to simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # Accept new connection
    print("Connection from: " + str(address))
    while True:
        # Receive data stream. It won't accept data packets greater than 1024 bytes
        data = conn.recv(1024).decode()
        if not data:
            # If data is not received, break
            break
        print("From connected user: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())  # Send data to the client

    conn.close()  # Close the connection

if __name__ == '__main__':
    server_program()
```

2.2 Client Side (client.py)



```
import socket

def client_program():
    host = socket.gethostname()  # As both code is running on the same PC
    port = 5000  # Socket server port number

    client_socket = socket.socket()  # Instantiate
    client_socket.connect((host, port))  # Connect to the server

    message = input(" -> ")  # Take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())  # Send message
        data = client_socket.recv(1024).decode()  # Receive response

        print('Received from server: ' + data)  # Show in terminal

        message = input(" -> ")  # Again take input

    client_socket.close()  # Close the connection

if __name__ == '__main__':
    client_program()
```

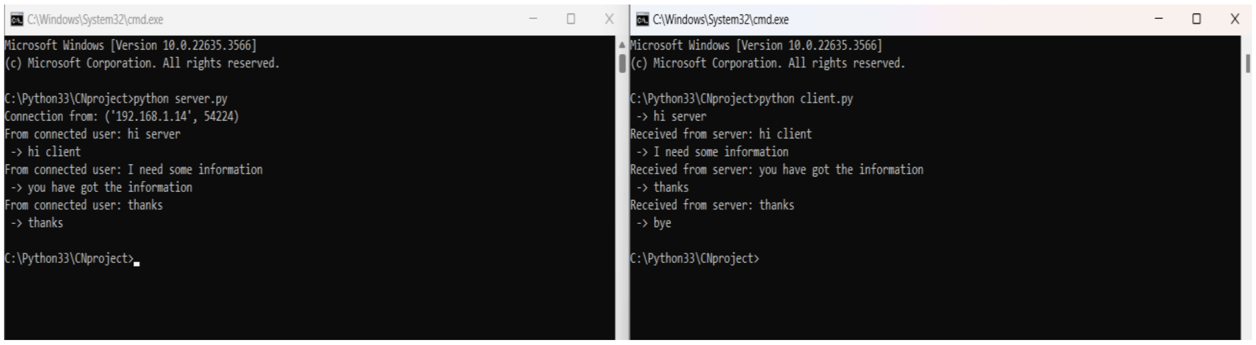
3. Explanation

Server Side: The server script binds to a hostname and port, listens for incoming connections, and accepts a client connection. It then enters a loop where it receives messages from the client, prints them, prompts for a response from the server side, and sends it back to the client.

Client Side: The client script connects to the server using the same hostname and port. It then enters a loop where it sends messages to the server, receives responses, and prints them. The loop continues until the user inputs 'bye'.

Results and Conclusion

Upon running both scripts, the client can send messages to the server, and the server can respond accordingly. This demonstrates a basic form of bidirectional communication between a server and a client using Python sockets. We can see results in the below screenshot where bidirectional communication can be seen.
Left is the server and the right side is the client side.



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.22635.3566]
(c) Microsoft Corporation. All rights reserved.

C:\Python33\CNproject>python server.py
Connection from: ('192.168.1.14', 54224)
From connected user: hi server
 -> hi client
From connected user: I need some information
 -> you have got the information
From connected user: thanks
 -> thanks

C:\Python33\CNproject>
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.22635.3566]
(c) Microsoft Corporation. All rights reserved.

C:\Python33\CNproject>python client.py
 -> hi server
Received from server: hi client
 -> I need some information
Received from server: you have got the information
 -> thanks
Received from server: thanks
 -> bye

C:\Python33\CNproject>
```