



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

A Lab Report
Of
Distributed System
On
Client-server Implementation

Submitted By:

Raj Kumar Dhakal (THA076BCT033)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

June, 2023

TITLE: SIMPLE CLIENT-SERVER IMPLEMENTATION USING PYTHON

THEORY:

A client-server network is a type of computer network architecture in which computing resources and services are distributed between clients and servers. In this network model, clients are devices or applications that request and consume services or resources, while servers are dedicated machines or software components that provide those services or resources.

The basic characteristics of a client-server network are as follows:

1. **Clients:** Clients are end-user devices or applications that interact with servers to access services or resources. Examples of clients include personal computers, laptops, smartphones, and web browsers. Clients initiate requests for specific services and receive responses from the servers.
2. **Servers:** Servers are dedicated machines or software programs designed to provide services or resources to clients. They are typically more powerful and have higher computing capabilities than clients. Servers listen for incoming requests from clients, process those requests, and send back the requested information or perform the requested actions.
3. **Communication:** Clients and servers communicate with each other over a network using communication protocols. Commonly used protocols include HTTP (Hypertext Transfer Protocol), TCP/IP (Transmission Control Protocol/Internet Protocol), and FTP (File Transfer Protocol). These protocols govern the format, rules, and procedures for transmitting data between clients and servers.

CODE:

Server.py

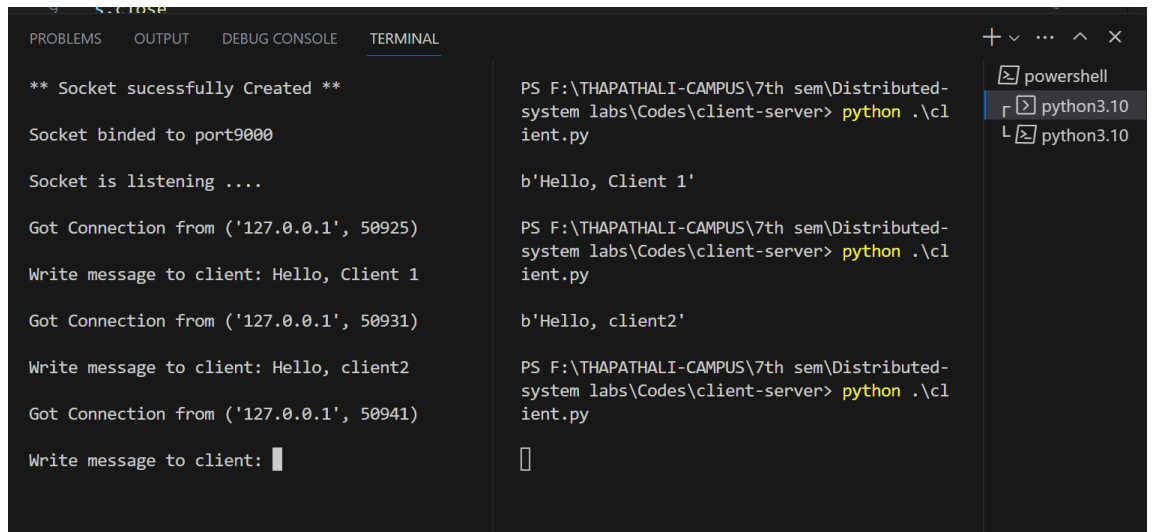
```
server.py  client.py
client-server > server.py > ...
1  import socket
2  s = socket.socket()
3  print("** Socket sucessfully Created ** \n")
4  port = 9000
5  s.bind(('',port)) #takes ip address and port
6  print(f'Socket binded to port{port} \n')
7  s.listen(6) #listen to atmost 6 connections
8  print('Socket is listening .... \n')
9  while True:
10     connection, addr =s.accept() #connection from client and it's address
11     print('Got Connection from', addr)
12     Message = input ("Write message to client: ") #Respond to the client
13     connection.send(Message.encode())
14     connection.close()
15
16
```

Client.py

```
server.py  client.py  X
client-server > client.py > ...
1  import socket
2  s = socket.socket()
3  port = 9000
4
5  s.connect(('127.0.0.1', port))
6  print(s.recv(1024))
7  s.close

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
```

OUTPUT:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
** Socket sucessfully Created **
Socket binded to port9000
Socket is listening ....
Got Connection from ('127.0.0.1', 50925)
Write message to client: Hello, Client 1
Got Connection from ('127.0.0.1', 50931)
Write message to client: Hello, client2
Got Connection from ('127.0.0.1', 50941)
Write message to client: █

PS F:\THAPATHALI-CAMPUS\7th sem\Distributed-system labs\Codes\client-server> python .\client.py
b'Hello, Client 1'
PS F:\THAPATHALI-CAMPUS\7th sem\Distributed-system labs\Codes\client-server> python .\client.py
b'Hello, client2'
PS F:\THAPATHALI-CAMPUS\7th sem\Distributed-system labs\Codes\client-server> python .\client.py
█
```

CONCLUSION:

Through the completion of this lab, I have acquired a comprehensive understanding of Python socket programming. I have familiarized myself with the different techniques employed in binding, accepting, and transmitting messages between the server and the client. This practical exercise has deepened my comprehension of the functioning principles underlying client-server architecture.