RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Name: Harshit Kumar
Reg No: 21ETMC412011

Experiment 7: Creating a TCP Socket-Based Chatroom

Aim: To use TCP Sockets to implement the TCP Socket-Based Chatroom
Objective: After carrying out this experiment, students will be able to:

- Apply TCP Socket programming technique to develop chatroom based on TCP sockets

Problem statement: You are required to write programs to implement a TCP based chatroom.

Analysis: While analyzing your program, you are required to address the following points:

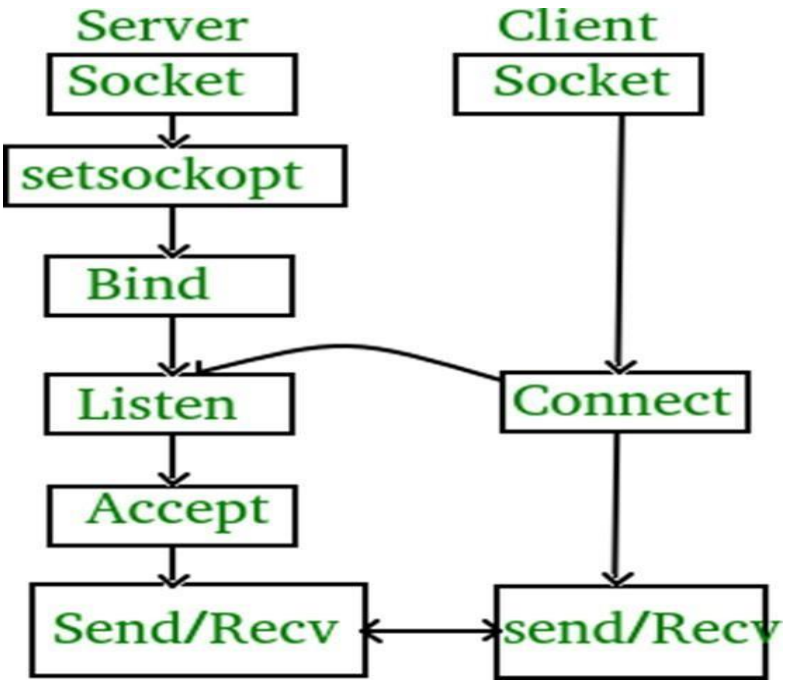
- How does the functionality of the program differ when you have the `accept()` function call at the server within an infinite loop as opposed to having it outside?

MARKS DISTRIBUTION:
Algorithm / Flowchart

Component	Maximum Marks	Marks Obtained
Preparation of Document	7	
Results	7	
Viva	6	
Total	**20**	

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Name: Harshit Kumar
Reg No: 21ETMC412011



Program:

Server.

File Edit Selection Find View Goto Tools Project Preferences Help

chart_server.py | chart_client.py

```
import time, socket, sys
print('Setup Server...')
time.sleep(1)
#Get the hostname, IP Address from socket and set Port
soc = socket.socket()
host_name = socket.gethostname()
ip = socket.gethostbyname(host_name)
port = 1234
soc.bind((host_name, port))
print(host_name, '({})'.format(ip))
name = input('Enter name: ')
soc.listen(1) #Try to locate using socket
print('Waiting for incoming connections...')
connection, addr = soc.accept()
print("Received connection from ", addr[0], "(", addr[1], ")\n")
print('Connection Established. Connected From: {}, ({})'.format(addr[0], addr[0]))
#get a connection from client side
client_name = connection.recv(1024)
client_name = client_name.decode()
print(client_name + ' has connected.')
print('Press [bye] to leave the chat room')
connection.send(name.encode())
while True:
    message = input('Me > ')
    if message == '[bye]':
        message = 'Bye, leaving the chat room!'
        connection.send(message.encode())
        print("\n")
        break
    connection.send(message.encode())
    message = connection.recv(1024)
    message = message.decode()
    print(client_name, '>', message)
```

Client.

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Name: Harshit Kumar
Reg No: 21ETMC412011

chart_client.py > ...

```python
import time, socket, sys
print('Client Server...')
time.sleep(1)
#Get the hostname, IP Address from socket and set Port
soc = socket.socket()
shost = socket.gethostname()
ip = socket.gethostbyname(shost)
#get information to connect with the server
print(shost, '({})'.format(ip))
server_host = input('Enter server\'s IP address:')
name = input('Enter Client\'s name: ')
port = 1234
print('Trying to connect to the server: {}, ({})'.format(server_host, port))
time.sleep(1)
soc.connect((server_host, port))
print("Connected...\n")
soc.send(name.encode())
server_name = soc.recv(1024)
server_name = server_name.decode()
print('{} has joined...'.format(server_name))
print('Enter [bye] to exit.')
while True:
    message = soc.recv(1024)
    message = message.decode()
    print(server_name, ">", message)
    message = input("Me > ")
    if message == "[bye]":
        message = "Leaving the Chat room"
        soc.send(message.encode())
        print("\n")
        break
    soc.send(message.encode())

```

OUTPUT:

Server.

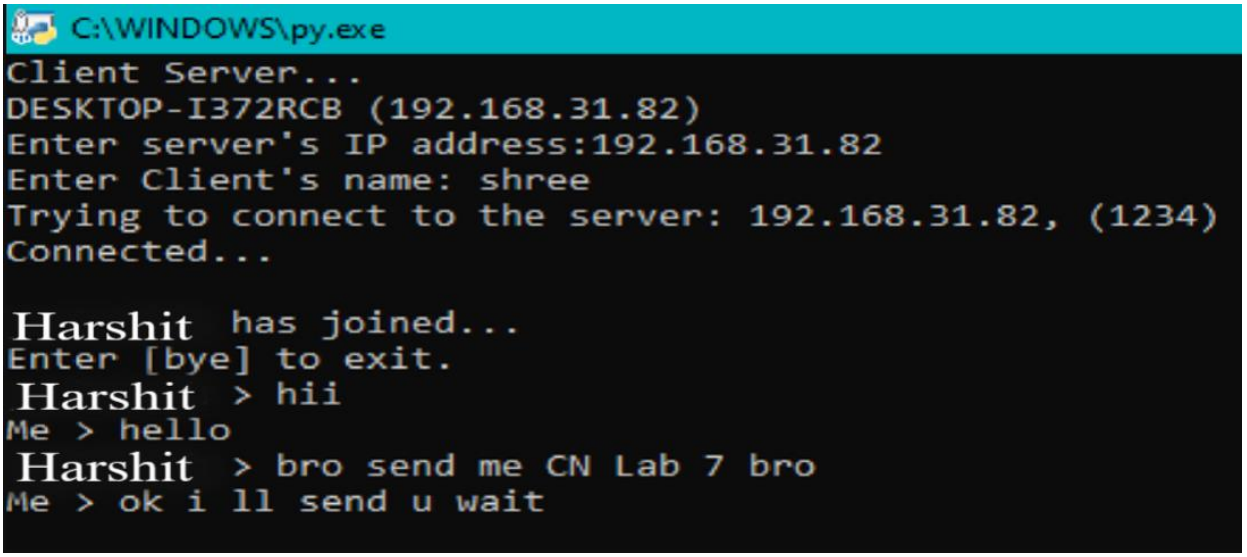
```
C:\WINDOWS\py.exe
DESKTOP-I372RCB (192.168.31.82)
Enter name:Harshit
Waiting for incoming connections...
Received connection from  192.168.31.82 ( 51456 )

Connection Established. Connected From: 192.168.31.82, (192.168.31.82)
shree has connected.
Press [bye] to leave the chat room
Me > hii
shree > hello
Me > bro send me CN Lab 7 bro
shree > ok i ll send u wait
Me > bye
shree > bye
Me > _
```

Client.

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Name: Harshit Kumar
Reg No: 21ETMC412011



Analysis and discussion:

Answer[1.]:

When the `accept() function call is within an infinite loop at the server:

The server continuously listens for and accepts incoming connections.
It can handle multiple client connections concurrently.
Each accepted connection spawns a new thread or process for communication.

When the `accept() function call is outside the infinite loop:

The server accepts only one connection and then exits.
It doesn't listen for further connections unless the program is restarted.
It's suitable for single-client applications or scenarios where the server doesn't need to handle multiple connections simultaneously.

Conclusion:

Understanding and implementing the technique of socket programming.