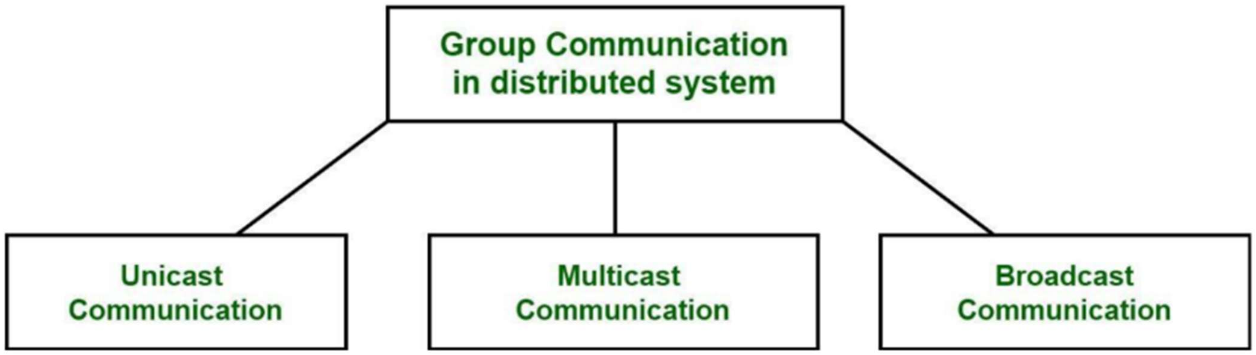
# Experiment 3

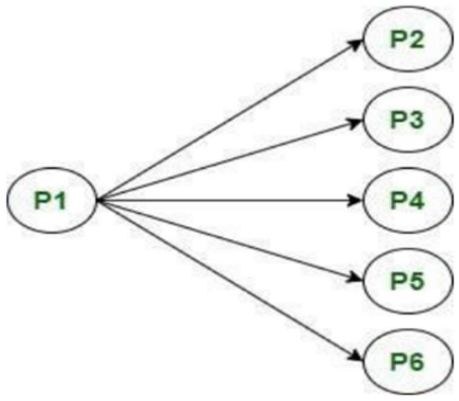
**Aim**: Write a program to implement Group Communication in a distributed system.

**Theory:** Communication between two processes in a distributed system is required to exchange various data, such as code or a file, between the processes. When one source process tries to communicate with multiple processes at once, it is called Group Communication. A group is a collection of interconnected processes with abstraction. This abstraction is to hide the message passing so that the communication looks like a normal procedure call. Group communication also helps the processes from different hosts to work together and perform operations in a synchronized manner, therefore increasing the overall performance of the system.



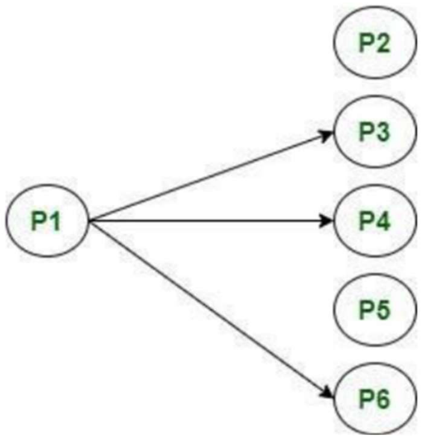
**Types of Group Communication in a Distributed System:**

Broadcast Communication: When the host process tries to communicate with every process in a distributed system at same time. Broadcast communication comes in handy when a common stream of information is to be delivered to each and every process in the most efficient manner possible. Since it does not require any processing whatsoever, communication is very fast in comparison to other modes of communication. However, it does not support a large number of processes and cannot treat a specific process individually.



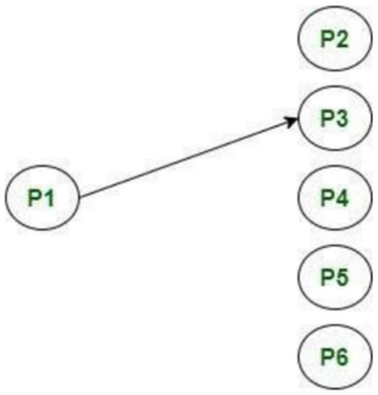
A broadcast Communication: P1 process communicating with every process in the system.

Multicast Communication: When the host process tries to communicate with a designated group of processes in a distributed system at the same time. This technique is mainly used to find a way to address the problem of a high workload on the host system and redundant information from processes in the system. Multitasking can significantly decrease time taken for message handling.



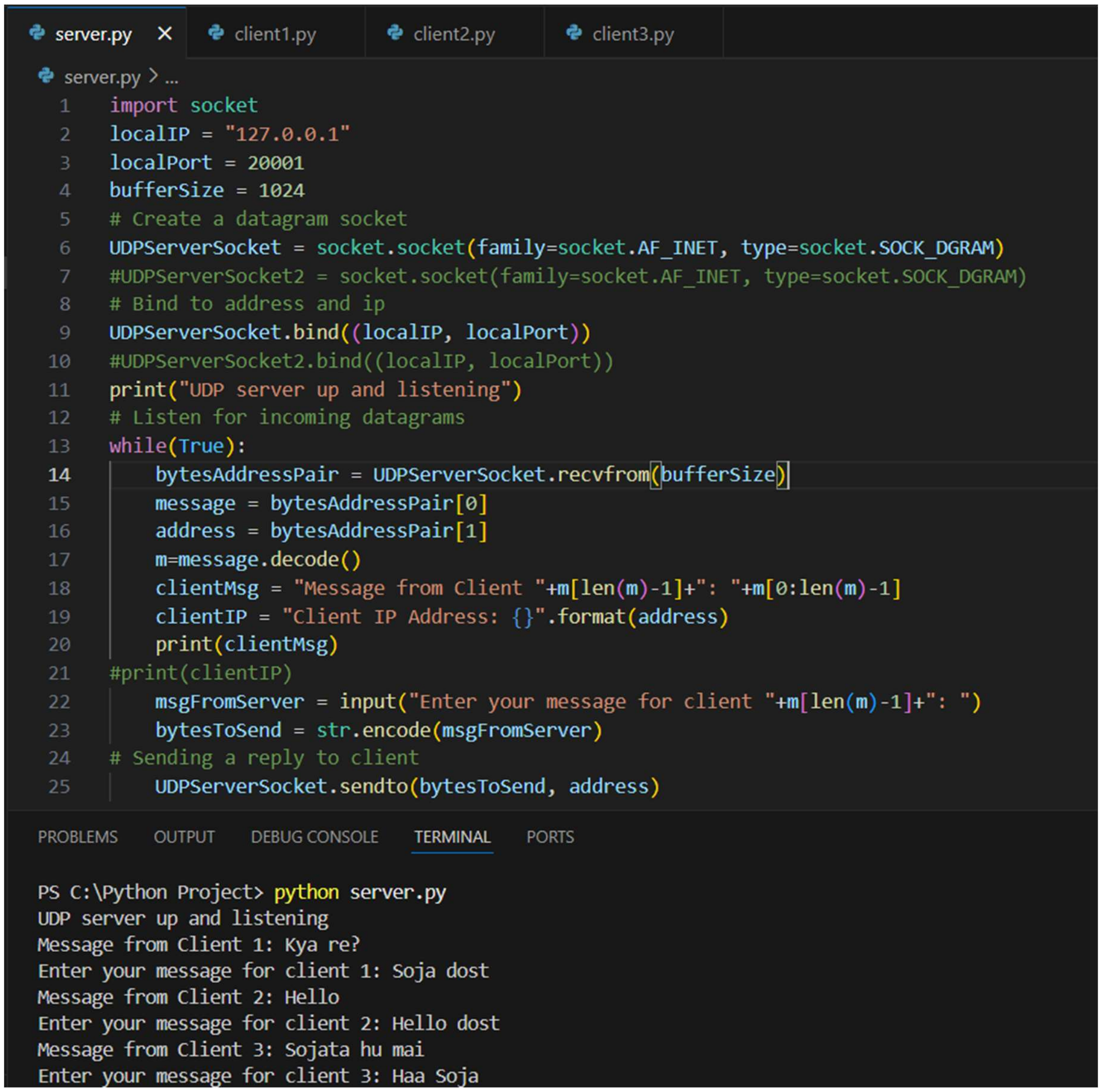
A multicast Communication: P1 process communicating with only a group of the process in the system

Unicast Communication: When the host process tries to communicate with a single process in a distributed system at the same time. Although, the same information may be passed to multiple processes. This works best for two processes communicating as only it has to treat a specific process only. However, it leads to overheads as it has to find the exact process and then exchange information/data.



A unicast Communication: P1 process communicating with only P3 process

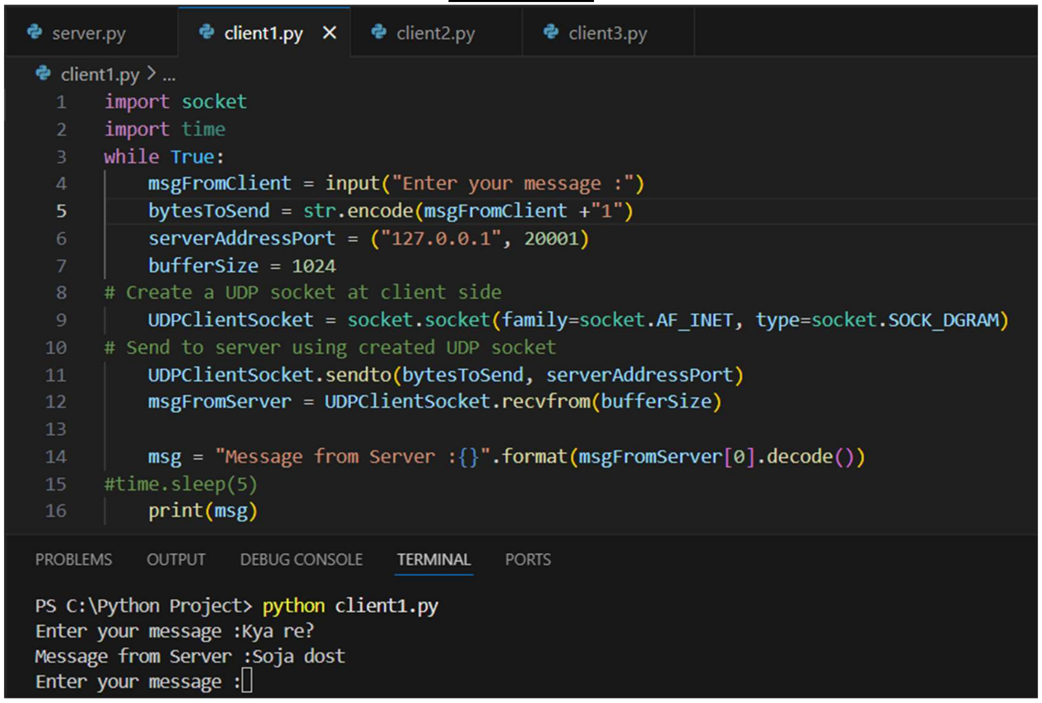
**Program and Output**: **server.py**



```python
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
# Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
#UDPServerSocket2 = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
#UDPServerSocket2.bind((localIP, localPort))
print("UDP server up and listening")
# Listen for incoming datagrams
while(True):
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
    message = bytesAddressPair[0]
    address = bytesAddressPair[1]
    m=message.decode()
    clientMsg = "Message from Client "+m[len(m)-1]+": "+m[0:len(m)-1]
    clientIP = "Client IP Address: {}".format(address)
    print(clientMsg)
#print(clientIP)
    msgFromServer = input("Enter your message for client "+m[len(m)-1]+": ")
    bytesToSend = str.encode(msgFromServer)
# Sending a reply to client
    UDPServerSocket.sendto(bytesToSend, address)
```

```
PS C:\Python Project> python server.py
UDP server up and listening
Message from Client 1: Kya re?
Enter your message for client 1: Soja dost
Message from Client 2: Hello
Enter your message for client 2: Hello dost
Message from Client 3: Sojata hu mai
Enter your message for client 3: Haa Soja
```

**client1.py**

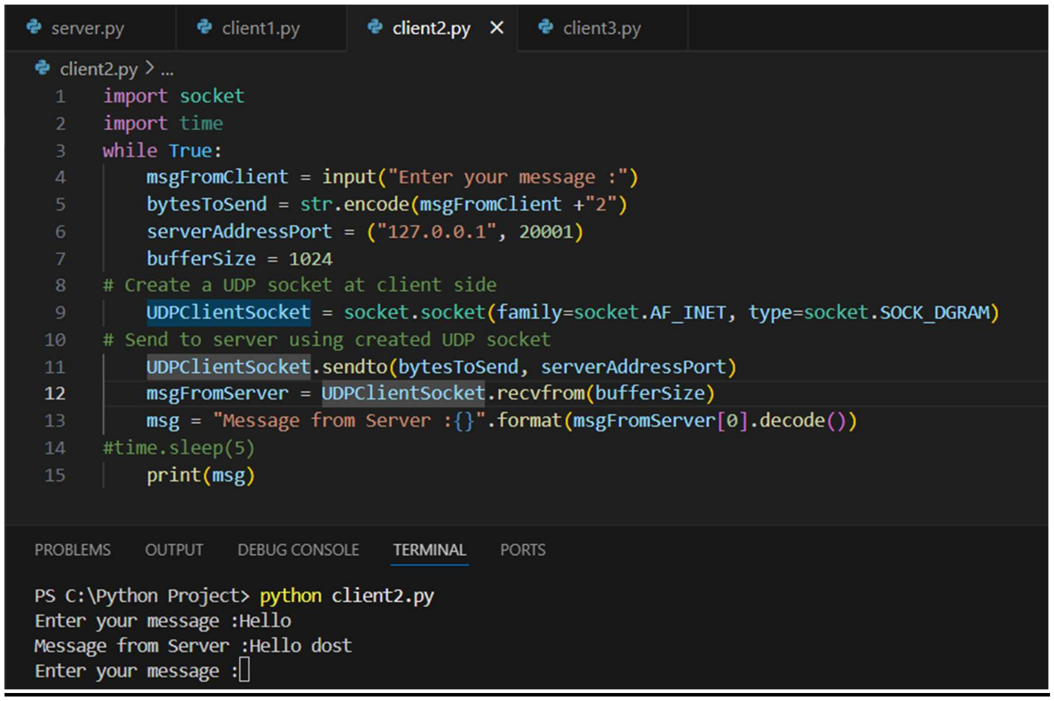


```python
import socket
import time
while True:
    msgFromClient = input("Enter your message :")
    bytesToSend = str.encode(msgFromClient +"1")
    serverAddressPort = ("127.0.0.1", 20001)
    bufferSize = 1024
# Create a UDP socket at client side
    UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Send to server using created UDP socket
    UDPClientSocket.sendto(bytesToSend, serverAddressPort)
    msgFromServer = UDPClientSocket.recvfrom(bufferSize)

    msg = "Message from Server :{}".format(msgFromServer[0].decode())
#time.sleep(5)
    print(msg)
```

```
PS C:\Python Project> python client1.py
Enter your message :Kya re?
Message from Server :Soja dost
Enter your message :
```

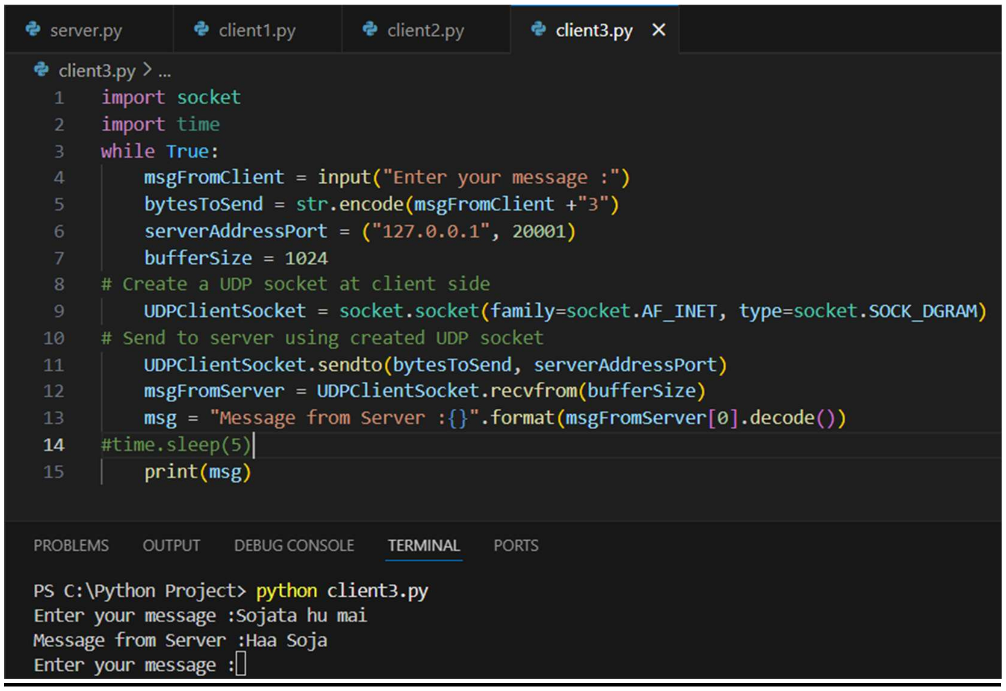
## client2.py



```python
import socket
import time
while True:
    msgFromClient = input("Enter your message :")
    bytesToSend = str.encode(msgFromClient +"2")
    serverAddressPort = ("127.0.0.1", 20001)
    bufferSize = 1024
# Create a UDP socket at client side
    UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Send to server using created UDP socket
    UDPClientSocket.sendto(bytesToSend, serverAddressPort)
    msgFromServer = UDPClientSocket.recvfrom(bufferSize)
    msg = "Message from Server :{}".format(msgFromServer[0].decode())
#time.sleep(5)
    print(msg)
```

```
PS C:\Python Project> python client2.py
Enter your message :Hello
Message from Server :Hello dost
Enter your message :
```

## client3.py



```python
import socket
import time
while True:
    msgFromClient = input("Enter your message :")
    bytesToSend = str.encode(msgFromClient +"3")
    serverAddressPort = ("127.0.0.1", 20001)
    bufferSize = 1024
# Create a UDP socket at client side
    UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Send to server using created UDP socket
    UDPClientSocket.sendto(bytesToSend, serverAddressPort)
    msgFromServer = UDPClientSocket.recvfrom(bufferSize)
    msg = "Message from Server :{}".format(msgFromServer[0].decode())
#time.sleep(5)
    print(msg)
```

```
PS C:\Python Project> python client3.py
Enter your message :Sojata hu mai
Message from Server :Haa Soja
Enter your message :
```

## Conclusion:

Thus, we have successfully implemented Group Communication in a distributed system.