Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai – 401208
Accredited A Grade by NAAC

Experiment 3

Name of the Student: -Rutuja Rajesh Kini
Roll No:-53
Date of Practical Performed: -5/2/25 Staff Signature with Date & Marks

Aim: Write a program to perform Group Communication.

Theory: Group communication refers to a communication paradigm where a message is sent to a group of processes rather than to an individual process. It is commonly used in systems where multiple processes need to collaborate or synchronize with each other. The goal is to manage the communication among distributed entities efficiently and correctly, ensuring that messages are delivered to all members of the group in a reliable, timely, and consistent manner.

Types of group communication:

1. **Unicast Communication** A type of communication where a message is sent from one sender to a specific receiver (a single process).
2. **Multicast Communication** A communication model where a message is sent from one sender to multiple receivers (a group of processes) at once. It is a more efficient version of sending messages individually (unicast) to each receiver.
3. **Broadcast Communication** Broadcast sends a message to **all** processes in a network or system, without targeting any specific group. In some cases, it can also be viewed as a form of multicast where the target group is the entire network.
4. **Anycast Communication** A type of communication where the message is sent to the nearest member of a group (based on some defined criteria, usually distance or latency), rather than to all members.

Modes of Communication in Group Communication:

1. **Synchronous Communication** In synchronous communication, the sender waits for an acknowledgment or response before proceeding further. The sender and receiver are synchronized in terms of timing.
2. **Asynchronous Communication** In asynchronous communication, the sender does not wait for an acknowledgment or response and proceeds immediately after sending the message. There is no guarantee about the timing of delivery or acknowledgment.
3. **Point-to-Point Communication** Point-to-point communication is a one-to-one interaction between two processes, where each message is directed from one sender to one receiver.

Code:

Server-

```
#Server
import socket
import threading

# Function to handle communication with each client
def handle_client(client_socket, client_address):
    print(f"New connection: {client_address}")
    while True:
        try:
            message = client_socket.recv(1024)
            if not message:
                break  # No more data from the client
            print(f"Message from {client_address}: {message.decode('utf-8')}")
            broadcast(message, client_socket)  # Send message to all clients
        except:
            break

    client_socket.close()
    print(f"Connection closed: {client_address}")

# Function to broadcast messages to all connected clients
def broadcast(message, sender_socket):
    for client in clients:
        if client != sender_socket:
            try:
                client.send(message)
            except:
                clients.remove(client)

# Set up the server socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(("0.0.0.0", 5555))  # Bind to all interfaces on port 5555
server_socket.listen(5)  # Listen for up to 5 incoming connections
clients = []
```

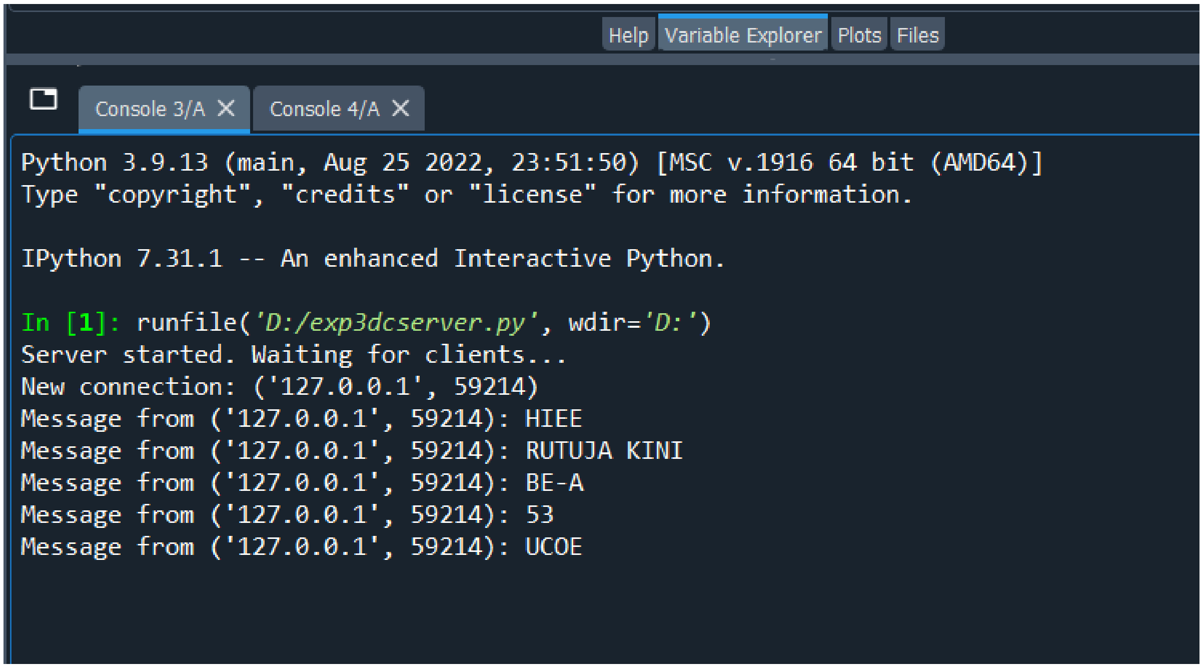
```
print("Server started. Waiting for clients...")

# Accept client connections and handle them in separate threads
while True:
    client_socket, client_address = server_socket.accept()
    clients.append(client_socket)
    threading.Thread(target=handle_client, args=(client_socket, client_address)).start()
```

Client-

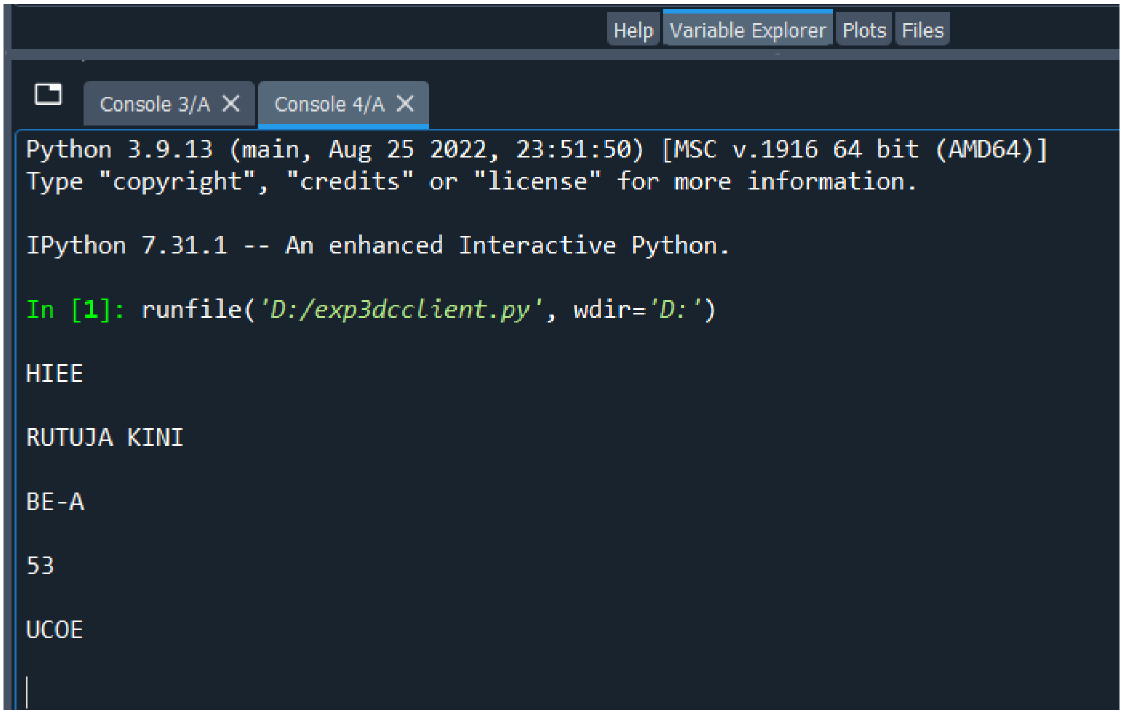
```
#client

# -*- coding: utf-8 -*-
"""
Created on Wed Feb  5 11:32:47 2025

@author: STUDENT
"""

import socket
import threading

# Function to receive messages from the server
def receive_messages(client_socket):
    while True:
        try:
            message = client_socket.recv(1024)
            print(f"\nNew message: {message.decode('utf-8')}")
        except:
            print("Connection lost.")
            break

# Set up the client socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(("127.0.0.1", 5555))  # Connect to the server

# Start the thread for receiving messages
threading.Thread(target=receive_messages, args=(client_socket,)).start()
```

```
# Send messages to the server
while True:
    message = input()
    if message:
        client_socket.send(message.encode('utf-8'))
```

Output:



```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('D:/exp3dcserver.py', wdir='D:')
Server started. Waiting for clients...
New connection: ('127.0.0.1', 59214)
Message from ('127.0.0.1', 59214): HIEE
Message from ('127.0.0.1', 59214): RUTUJA KINI
Message from ('127.0.0.1', 59214): BE-A
Message from ('127.0.0.1', 59214): 53
Message from ('127.0.0.1', 59214): UCOE
```



```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('D:/exp3dcclient.py', wdir='D:')

HIEE

RUTUJA KINI

BE-A

53

UCOE
```

Conclusion: Thus, we have implemented group communication.