Experiment-8

Aim: Write a program for interactive application using UDP socket.

Prerequisite: Nil

Outcome: To impart knowledge of Socket Programming

Theory:

- Socket programming serves as a foundational element in computer networking, enabling communication between different devices or applications over a network, be it local or spanning considerable distances. It allows programs to transmit and receive data across network connections.
- In this framework, a "socket" stands as the endpoint for communication between two machines. Each socket is characterized by an associated IP address and port number, uniquely identifying a specific process or service on a device. Sockets operate in either client mode, initializing the connection, or server mode, awaiting incoming connections. Once connected, data can flow bidirectionally.
- Socket programming involves utilizing a suite of functions offered by the operating system or programming libraries to create, configure, and manage these communication endpoints. Widely used programming languages like Python, Java, C/C++, and others furnish libraries or modules simplifying the socket programming process. This accessibility broadens its use, empowering diverse developers to craft a range of networked applications, including web servers, chat platforms, multiplayer games, and more.

Procedure:

1. Write Simple Client Server Program using Java/Python Programming Language.
2. Execute the program using appropriate compiler.
3. Verify the working of the program.

Steps:

Server.py

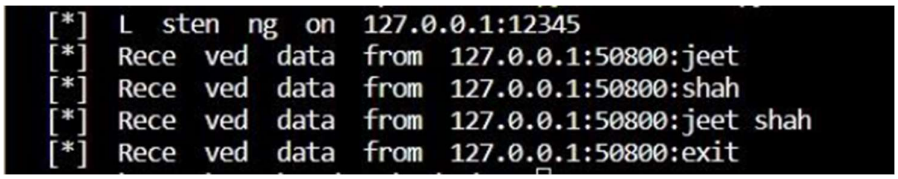
```
import socket

# Define server address (Ifl and port)
SERVER_HOST = '127.0.0.1'
SERVER_flORT = 12345

# Create a UDfl socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to the address
server_socket.bind((SERVER_HOST, SERVER_flORT))
# fprint a message indicating that the server is listening
print(f"[*] L sten ng on {SERVER_HOST}:{SERVER_flORT}")

# Receive and process incoming messages
while True:
data,client_address = server_socket.recvfrom( 1024) #
Receive data from client
message = data.decode() #Decode rece ved data to string
print(f"[*] Rece ved data from {client_address[0]}:{client_address[1]}
:{message}")
 if message.lower() == 'exit':
 break #Exit the loop if 'exit' is received

# Close the server socket
server_socket.close()
```

Client.py

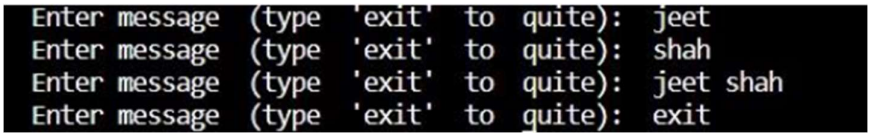
```
import socket

# Define server address (Ifl and port)

SERVER_HOST = '127.0.0.1'

SERVER_flORT = 12345

# Create a UDfl socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Send messages to the server

while True:

message = input("Enter message (type 'exit' to quite): ")

# Send data to the server

client_socket.sendto(message.encode(), (SERVER_HOST, SERVER_flORT))

if message.lower() == 'exit':

break #Exit the loop if 'exit' is sent

# Close the client socket

client_socket.close()
```

Output:

Server Slide



Client Slide



Observation & Learning

- Socket programming lays the groundwork for inter-device communication, enabling the exchange of data among devices, such as clients and servers. In a specific case, a server script (server.py) was established using Python's socket library. This server was set to listen on IP 127.0.0.1 and port 12345, ready to accept incoming connections.
- A corresponding client script (client.py) was designed to connect to the server using the same IP and port. It permitted users to input messages for the server and offered an option to exit. The server was programmed to mirror back any messages it received from the client, showcasing the smooth transfer of data between them.

Conclusion

In conclusion, this experiment offered significant insights into UDP socket programming and the creation of simple network applications. It was evident that both the server and client effectively initiated connections, sent, and received messages, illustrating the fundamental principles of communication using UDP. This practical experience forms a strong base for comprehending more intricate network applications and protocols that rely on UDP.