Government College of Engineering, Jalgaon

Department of Computer Engineering

(CO309-Computer Network Technology Lab)

Name: Pallavi Govardhan Patil. PRN: 2241046. Class: TY.

Practical No: 1. Semester: V Subject Teacher: Ms. Prajakta Sawle.

Date of Performance: Date Of completion:

Write Up	Correctness of Program	Documentation of Program	Viva	Attendance for Practical	Timely Completion	Total	Dated sign of Subject Teacher
4	2	2	5	2	5	20	

Practical No.1

❖ **AIM :-**

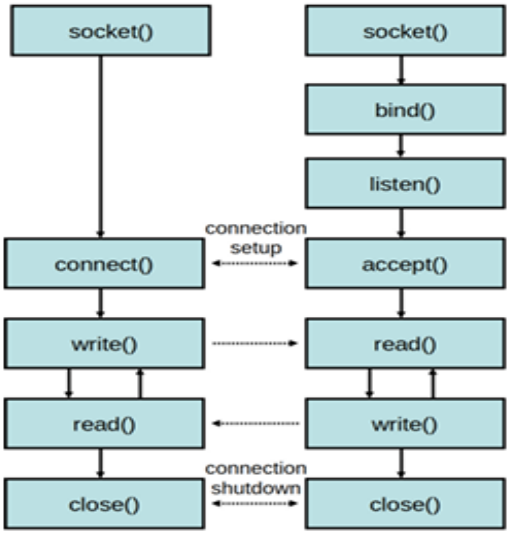
1. UNIX Sockets: WAP program in C/C++ /Python/Java sockets API.
   a. TCP sockets.
   b. UDP sockets.

❖ **SOFTWARE REQUIRED :-**

Operating System: - Ubuntu , Python3.12 .

❖ **THEROY:-**

Creating UDP server :-



1**. socket()**:

- Similar to the client, the server begins by creating a socket using the socket() function

2. **connect()**:

- The server binds the socket to a specific IP address and port using the bind() function.
- This step is crucial because it tells the operating system to listen for incoming data on this specific address and port.

3. **read() / write()**:

- The server reads data from the client using the read() function and sends data using the write() function. These operations are performed on the socket descriptor returned by accept().

4. **close()**:
    - Finally, when the communication is complete, the server closes the socket using the close() function, terminating the connection.

Creating a UDP Client :-

1. **socket()**:

    - The client begins by creating a socket using the socket() system call.
    - A socket is essentially an interface for network communication. It allows the client to send and receive data.
    - In UDP, a socket is created by specifying the Internet Protocol (IP) family (IPv4/IPv6), the type of communication (datagram-based for UDP), and the protocol to be used (UDP in this case).

2. **connect()**:

    - In a traditional TCP connection, this function would establish a reliable connection to a server. However, in the case of UDP, this step doesn't establish a persistent connection.
    - It simply specifies the destination (the server's IP address and port number), so the client knows where to send the data when using sendto().
    - The client now has the server's address information stored for future communication.

3. **Sendto()**:

    - Now that the client has the server's IP and port information, it sends data (a datagram) to the server using the sendto() function.
    - The client doesn't wait for a connection acknowledgment since UDP is connectionless; it simply sends the data packet and expects a response.

4. **Recvfrom():**

    - After sending the datagram, the client waits for a reply from the server.
    - The recvfrom() function is used to receive the server's response
    - This function will block (pause) until a datagram is received from the server.

5.read() / write():

    - The server reads data from the client using the read() function and sends data using the write() function. These operations are performed on the socket descriptor returned by accept().

6. Close():

    - Once the client has received the response and no further communication is required, it closes the socket using the close() function.

Conclusion :-

UDP enables fast, connectionless communication between a client and server by sending independent datagrams without establishing a persistent connection. It's efficient for speed-critical applications, though it doesn't guarantee reliable delivery.

Name & Sign of Course Teacher
Mrs.Prajakta DSawle