Socket Programming

Aim: To develop any distributed application through implementing client-server communication programs based on Java Sockets.

Related Theory:

Socket: In distributed computing, network communication is one of the essential parts of any system, and the socket is the endpoint of every instance of network communication. In Java communication, it is the most critical and basic object involved.

A socket is a handle that a local program can pass to the networking API to connect to another machine. It can be defined as the terminal of a communication link through which two programs /processes/threads running on the network can communicate with each other. The TCP

layer can easily identify the application location and access information through the port number assigned to the respective sockets.

During an instance of communication, a client program creates a socket at its end and tries to connect it to the socket on the server. When the connection is made, the server creates a socket at its end and then server and client communication is established.

Designing the solution:

The java.net package provides classes to facilitate the functionalities required for networking. The socket class programmed through Java using this package has the capacity of being independent of the platform of execution; also, it abstracts the calls specific to the operating system on which it is invoked from other Java interfaces. The ServerSocket class offers to observe connection invocations, and it accepts such invocations from different clients

through another socket. High-level wrapper classes, such as URLConnection and URLEncoder, are more appropriate. If you want to establish a connection to the Web using a URL, then these classes will use the socket internally.

The java.net package provides support for the two common network protocols −

TCP − TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.

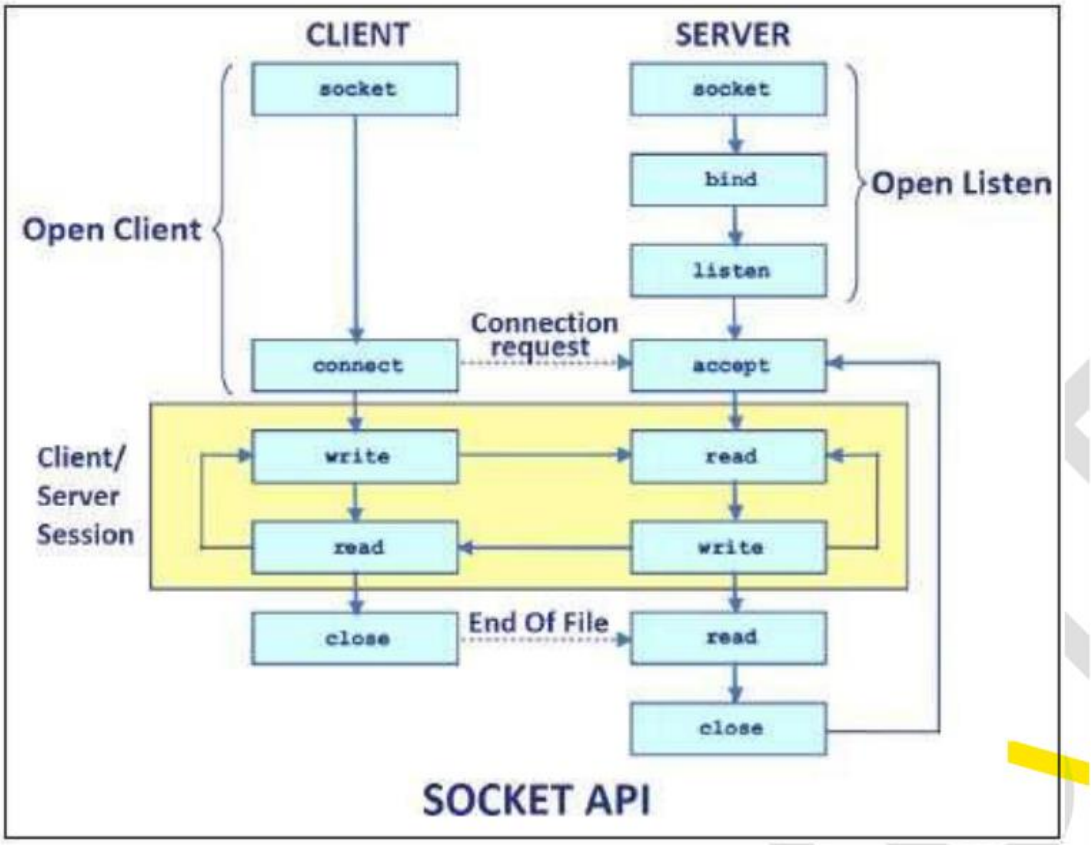
UDP − UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

Socket programming for TCP:

The following steps occur when establishing a TCP connection between two computers using sockets −

- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and the port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.
- After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.
- TCP is a two-way communication protocol, hence data can be sent across both streams at the same time. Following are the

useful classes providing complete set of methods to implement sockets.



Socket programming for UDP:

UDP is used only when the entire information can be bundled into a single packet and there is no dependency on the other packet. Therefore, the usage of UDP is quite limited, whereas TCP is widely used in IP applications. UDP sockets are used where limited bandwidth is available, and the overhead associated with resending packets is not acceptable.

To connect using a UDP socket on a specific port, use the following code:

DatagramSocket udpSock = new DatagramSocket(3000);

A datagram is a self-contained, independent message whose time of arrival, confirmation of arrival over the network, and content cannot be

guaranteed.datagramPacket objects are used to send data over DatagramSocket. Every DatagramPacket object consists of a data buffer, a remote host to whom the data needs to be sent, and a port number on which the remote agent would be listened.

Implementing the solution:

Socket Programming for TCP:

Client Programming:

1. Establish a Socket Connection: The java.net Socket class represents a Socket. To open a socket:

Socket socket = new Socket("127.0.0.1", 5000)

2. Communication: To communicate over a socket connection, streams are used to both input and output the data.

3. Closing the connection: The socket connection is closed explicitly once the message to server is sent.

Server Programming:

1. Establish a Socket Connection: To write a server application two sockets are needed. A ServerSocket which waits for the client requests (when a client makes a new Socket()). A plain old Socket socket to use for communication with the client.

2. Communication: getOutputStream() method is used to send the output through the socket.

3. Close the Connection: After finishing, it is important to close the connection by closing the socket as well as input/output streams.

Compilation and Executing the solution:

If you're using Eclipse :

1. Compile both of them on two different terminals or tabs

2. Run the Server program first

3. Then run the Client program

4. Type messages in the Client Window which will be received and showed by the Server.

5. Close the socket connection by typing something like "Exit".