

Sockets

Explicación teórica:

- **Definición:**

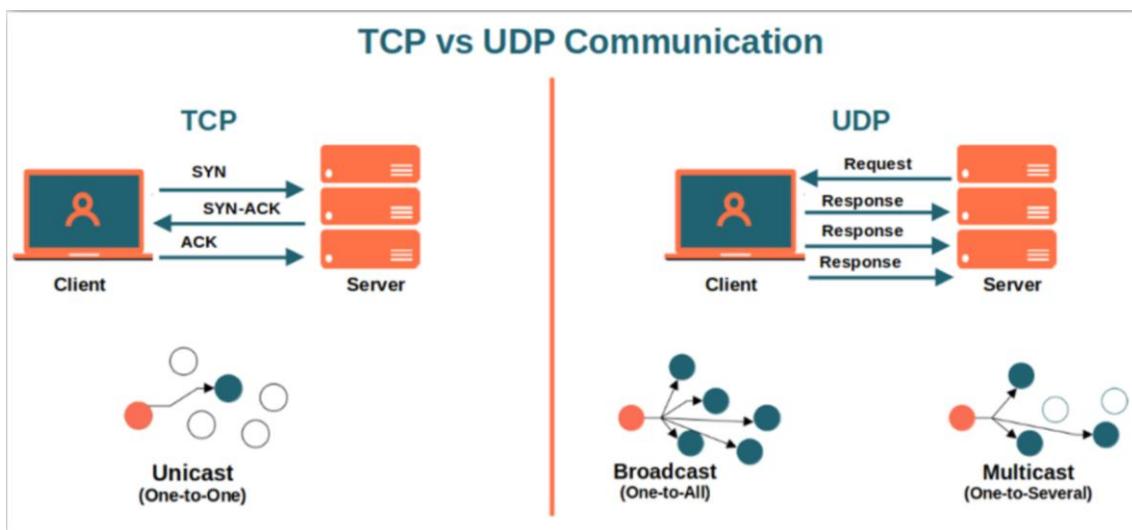
Un **socket** es un punto de conexión entre dos dispositivos que se comunican a través de una red. Se utiliza para enviar y recibir datos.

- **Tipos de Sockets:**

- **TCP (Transmission Control Protocol):** Protocolo orientado a conexión, fiable y garantiza la entrega de datos en orden.
- **UDP (User Datagram Protocol):** Protocolo sin conexión, rápido, pero no garantiza la entrega ni el orden.

- **Ejemplos de aplicaciones:**

- TCP: Navegadores web, FTP.
- UDP: Streaming, videojuegos en línea.



Clases Principales de Java Para trabajar con sockets

TCP

Diagrama de flujo:

- **Cliente → (conexión) → Servidor**
- **Cliente → (envía mensaje) → Servidor**
- **Servidor → (responde) → Cliente**

ServerSocket (Servidor TCP) Escucha conexiones entrantes en un puerto.

Métodos:

accept(): Acepta una conexión y devuelve un Socket.

close(): Cierra el servidor.

Ej/

```
ServerSocket serverSocket = new ServerSocket(15555);
Socket socket = serverSocket.accept(); // Espera una conexión
```

Socket (Cliente/Servidor TCP) Representa una conexión entre dos dispositivos.

- Métodos:

- **getInputStream()**: Flujo de entrada.
- **getOutputStream()**: Flujo de salida.

Ej/

```
Socket socket = new Socket("localhost", 12345);
```

Clases para UDP

- **DatagramSocket** Se utiliza para enviar y recibir paquetes de datos.
- Métodos:
 - send(DatagramPacket): Envía un paquete.
 - receive(DatagramPacket): Recibe un paquete.

```
DatagramSocket socket = new DatagramSocket(12345); // Puerto para recibir
```

DatagramPacket Representa un paquete de datos.

- Constructores:

- Para enviar:

```
DatagramPacket packet = new DatagramPacket(data, data.length,
address, port);
```

Para recibir:

```
DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
```

Paquetes en UDP

- Los datos se envían como bloques (byte[]) encapsulados en un DatagramPacket.

Ejemplo

Servidor TCP

```
import java.io.*;
import java.net.*;

public class ServidorTCP{
    public static void main(String[] args){
        int puerto = 12345; // Puerto del servidor

        try (ServerSocket serverSocket = new ServerSocket(puerto)){
            System.out.println("Servidor TCP escuchando en el puerto " + puerto);

            Socket socket = serverSocket.accept(); // Espera una conexión
            System.out.println("Cliente conectado.");

            BufferedReader in = new BufferedReader(new
                InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            String mensaje = in.readLine(); // Recibir mensaje
            System.out.println("Mensaje recibido: " + mensaje);

            out.println("Servidor: Recibí tu mensaje -> " + mensaje); // Responder

            socket.close();
        } catch (IOException e){
            System.out.println("Error en el servidor TCP: " + e.getMessage());
        }
    }
}
```

Cliente TCP

```
import java.io.*;
import java.net.*;

public class ClienteTCP {
    public static void main(String[] args) {
        String host = "localhost"; // Dirección del servidor
        int puerto = 12345;      // Puerto del servidor

        try (Socket socket = new Socket(host, puerto)) {
            System.out.println("Conectado al servidor TCP.");

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            out.println("Hola, servidor TCP!"); // Enviar mensaje
            String respuesta = in.readLine(); // Leer respuesta
            System.out.println("Respuesta del servidor: " + respuesta);

        } catch (IOException e) {
            System.out.println("Error en el cliente TCP: " + e.getMessage());
        }
    }
}
```