



19 de octubre

Comunicación UDP

Cliente-Servidor en Java

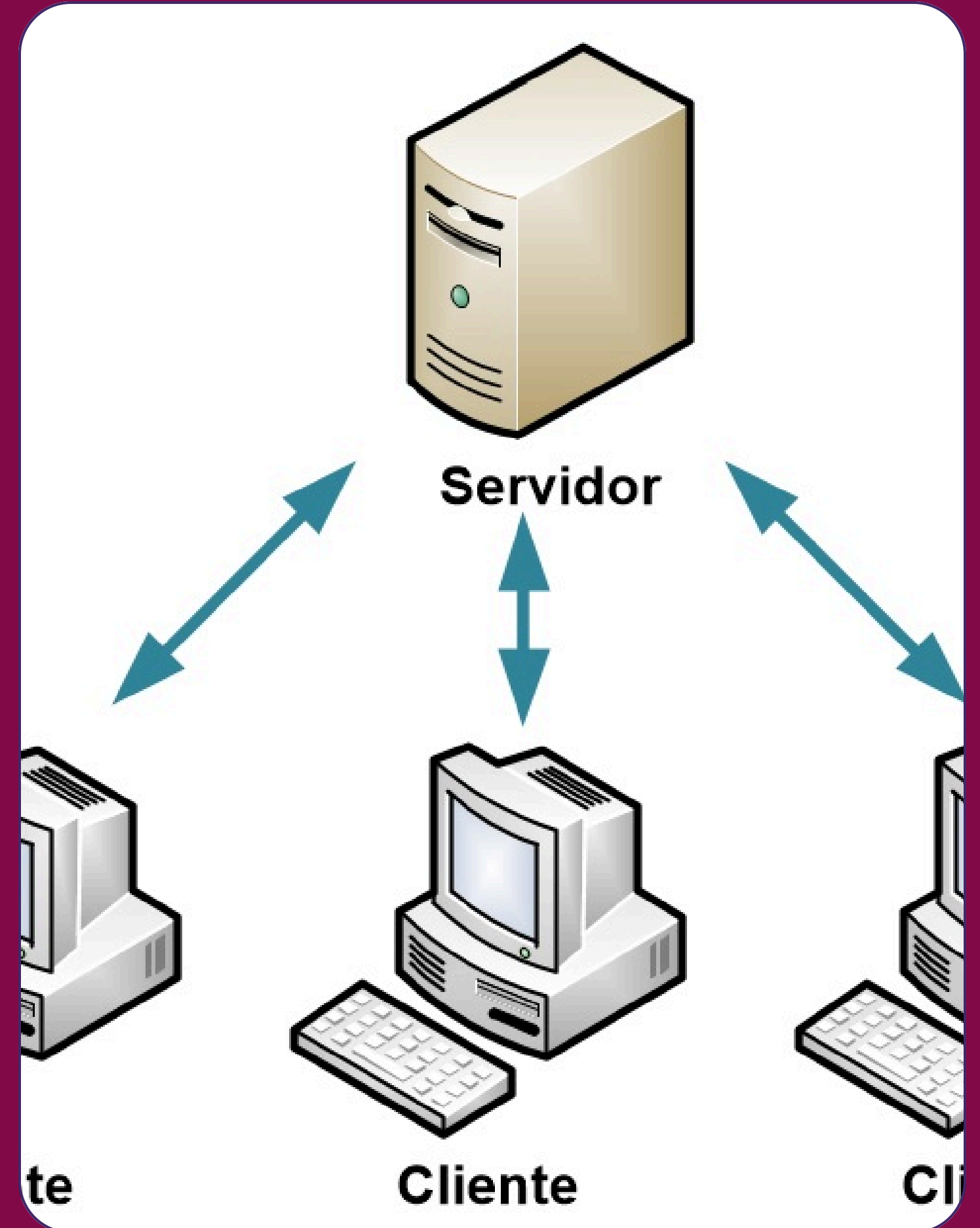


Instituto Universitario de Yucatán

PERLA JUDITH ARIAS REYES

Título e Introducción

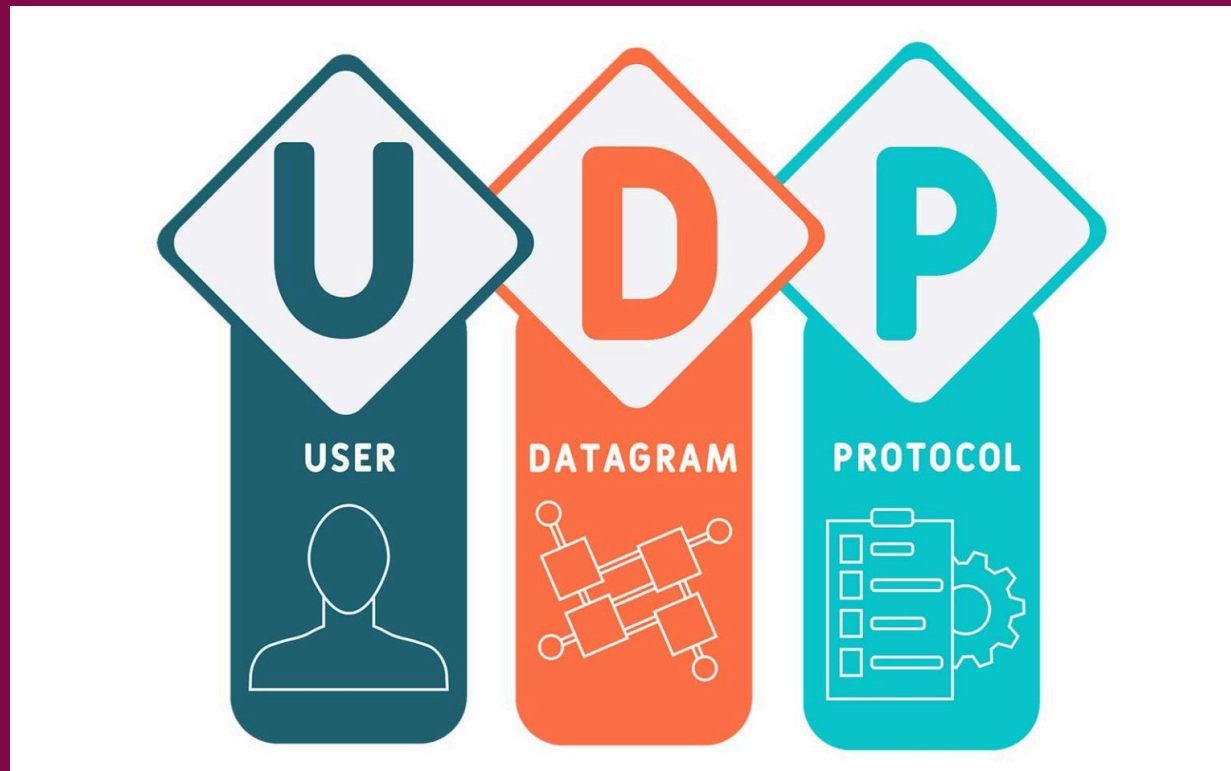
Esta presentación aborda la **comunicación UDP** entre cliente y servidor utilizando Java en un contexto educativo.



¿Qué es UDP?

Conceptos fundamentales del Protocolo UDP

UDP es un protocolo de comunicación **rápido y sin conexión** que permite el envío de datagramas sin confirmar la recepción, siendo eficiente para aplicaciones en tiempo real.



```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class ServidorUDP {
    private DatagramSocket socket;
    private final int PUERTO;
    private int mensajesRecibidos = 0;

    public ServidorUDP(int puerto) throws IOException {
        this.PUERTO = puerto;
        this.socket = new DatagramSocket(PUERTO);
    }
}
```


Flujo de Comunicación

Servidor inicia

El servidor **espera mensajes** en un puerto fijo.

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class ClienteUDP {
    private DatagramSocket socket;
    private InetAddress direccionServidor;
    private final int PUERTO_SERVIDOR = 9876;

    public ClienteUDP(String host) throws IOException {
        this.socket = new DatagramSocket();
        this.direccionServidor = InetAddress.getByName(host);
        System.out.println("🟢 Cliente UDP iniciado");
    }

    public void enviarMensaje(String mensaje) throws IOException {
        // Enviar mensaje al servidor
        byte[] buffer = mensaje.getBytes();
        DatagramPacket paquete = new DatagramPacket(
            buffer,
            buffer.length,
            direccionServidor,
            PUERTO_SERVIDOR
        );
        socket.send(paquete);
        System.out.println("📤 Enviado: " + mensaje);

        // Recibir respuesta del servidor
    }
}
```

Cliente envía

El cliente envía **mensajes al servidor** para recibir respuestas.

Servidor responde

El servidor **responde a cada mensaje** recibido del cliente.

Configuración del Servidor UDP en Java

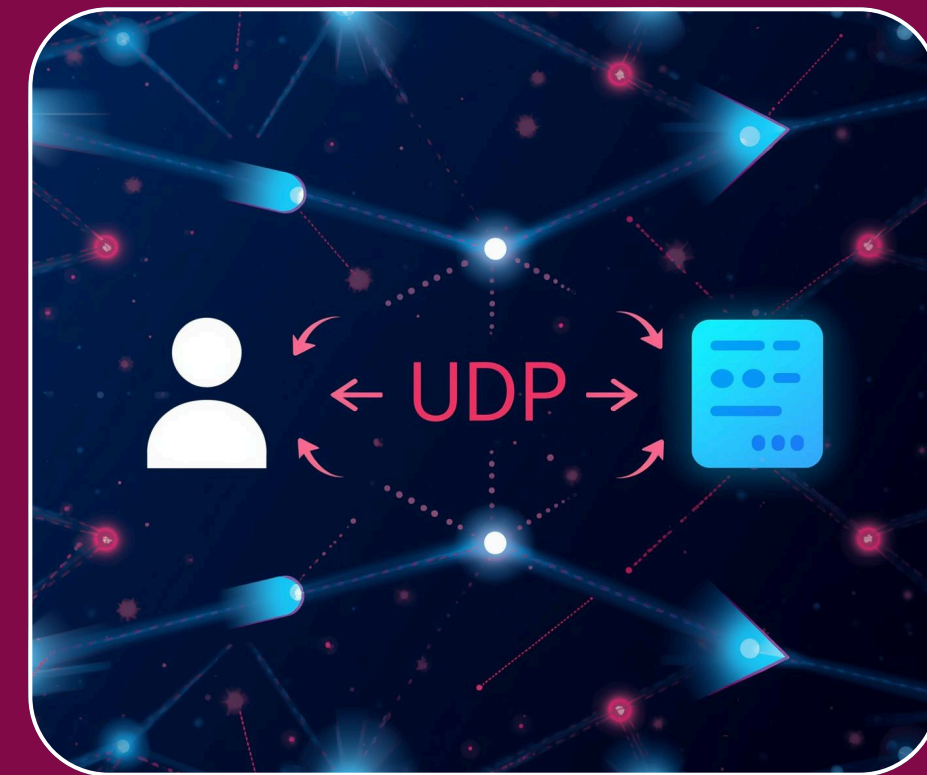
```
source History
1 import java.io.IOException;
2 import java.net.DatagramPacket;
3 import java.net.DatagramSocket;
4 import java.net.InetAddress;
5
6 public class ClienteUDP {
7     private DatagramSocket socket;
8     private InetAddress direccionServidor;
9     private final int PUERTO_SERVIDOR = 9876;
10
11     public ClienteUDP(String host) throws IOException {
12         this.socket = new DatagramSocket();
13         this.direccionServidor = InetAddress.getByName(host);
14         System.out.println("● Cliente UDP iniciado");
15     }
16
17     public void enviarMensaje(String mensaje) throws IOException {
18         // Enviar mensaje al servidor
19         byte[] buffer = mensaje.getBytes();
20         DatagramPacket paquete = new DatagramPacket(
21             buffer,
22             buffer.length,
23             direccionServidor,
24             PUERTO_SERVIDOR
25         );
26         socket.send(paquete);
27     }
28 }
```

Creación de Socket

El servidor inicia creando un socket en el puerto 9876.

Bucle Infinito

El servidor permanece en un ciclo infinito esperando mensajes.



Recepción de Datos

Utiliza un buffer para recibir datos continuos del cliente.

Manejo de Mensajes Entrantes y Respuesta

Recepción de Mensajes

El servidor recibe mensajes de manera **bloqueante**.

```
5 public void iniciar() throws IOException {
6     System.out.println("● Servidor UDP iniciado en puerto " + PUERTO);
7     System.out.println("⌂ Esperando mensajes...\n");
8
9     while (true) {
10         byte[] buffer = new byte[1024];
11         DatagramPacket paquete = new DatagramPacket(buffer, buffer.length);
12
13         // Recibir mensaje
14         socket.receive(paquete);
15         mensajesRecibidos++;
16
17         String mensaje = new String(paquete.getData(), 0, paquete.getLength());
18         String direccionCliente = paquete.getAddress().getHostAddress();
19         int puertoCliente = paquete.getPort();
20
21         System.out.println("✉ Mensaje #" + mensajesRecibidos + " recibido");
22         System.out.println("    De: " + direccionCliente + ":" + puertoCliente);
23         System.out.println("    Contenido: " + mensaje);
24
25         // Crear y enviar respuesta
26         String respuesta = "Servidor confirmó recepción del mensaje: '" + mensaje + "'";
27         byte[] bufferRespuesta = respuesta.getBytes();
28         DatagramPacket paqueteRespuesta = new DatagramPacket(
29             bufferRespuesta,
30             bufferRespuesta.length,
31             paquete.getAddress(),
32             paquete.getPort());
33     }
34 }
```

Procesamiento del Mensaje

Obtiene dirección y puerto del **cliente**.

Envío de Respuesta

Responde al cliente usando datos **obtenidos**.

Funcionamiento del Cliente UDP en Java

Crear Socket

El cliente genera un socket asignado aleatoriamente.

Obtener IP

Se conecta a la dirección IP del servidor.

Preparar Mensaje

Mensaje convertido a bytes para ser enviado.

```
public void enviarMensaje(String mensaje) throws IOException {
    // Enviar mensaje al servidor
    byte[] buffer = mensaje.getBytes();
    DatagramPacket paquete = new DatagramPacket(
        buffer,
        buffer.length,
        direccionServidor,
        PUERTO_SERVIDOR
    );
    socket.send(paquete);
    System.out.println("📤 Enviado: " + mensaje);

    // Recibir respuesta del servidor
    byte[] bufferRespuesta = new byte[1024];
    DatagramPacket paqueteRespuesta = new DatagramPacket(
        bufferRespuesta,
        bufferRespuesta.length
    );
    socket.receive(paqueteRespuesta);

    String respuesta = new String(
        paqueteRespuesta.getData(),
        0,
        paqueteRespuesta.getLength()
    );
    System.out.println("📥 Respuesta del servidor: " + respuesta);
    System.out.println("-----");
}
```

Recepción y Procesamiento de Respuestas

Espera activa

El cliente espera la respuesta del servidor.

Conversión de datos

Transformar los bytes en texto legible.

Visualización

Mostrar la respuesta en pantalla al usuario.

```
ServidorUDP (run) X  ServidorUDP (run) #2 X
run:
? Servidor UDP iniciado en puerto 9876
? Esperando mensajes...

? Mensaje #1 recibido
  De: 127.0.0.1:55259
  Contenido: Hola servidor!
? Respuesta enviada al cliente
-----

? Mensaje #2 recibido
  De: 127.0.0.1:55259
  Contenido: ¿Cómo estás?
? Respuesta enviada al cliente
-----
```

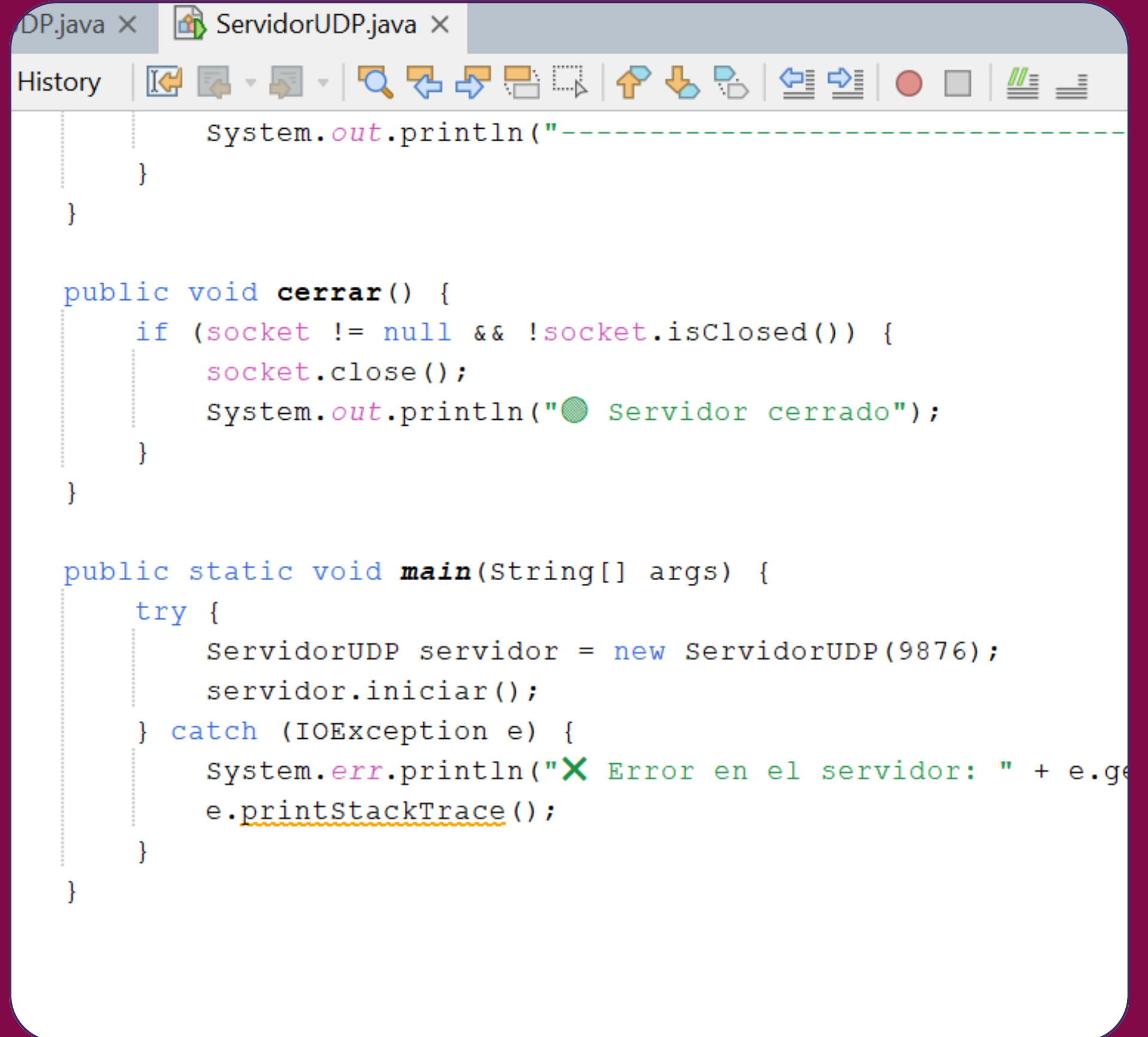
```
-----

? Mensaje #3 recibido
  De: 127.0.0.1:55259
  Contenido: Este es el tercer mensaje
? Respuesta enviada al cliente
-----

? Mensaje #4 recibido
  De: 127.0.0.1:55259
  Contenido: Enviando datos importantes
? Respuesta enviada al cliente
-----
```


Operaciones Bloqueantes

Las operaciones bloqueantes en UDP implican que el cliente y servidor esperan sin consumir CPU, optimizando recursos durante la comunicación.



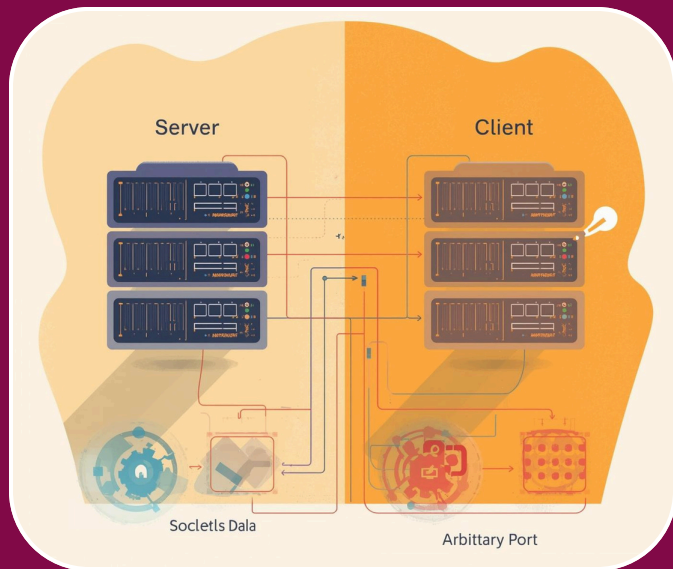
```
DP.java x ServidorUDP.java x
History

        System.out.println("-----");
    }
}

public void cerrar() {
    if (socket != null && !socket.isClosed()) {
        socket.close();
        System.out.println("● Servidor cerrado");
    }
}

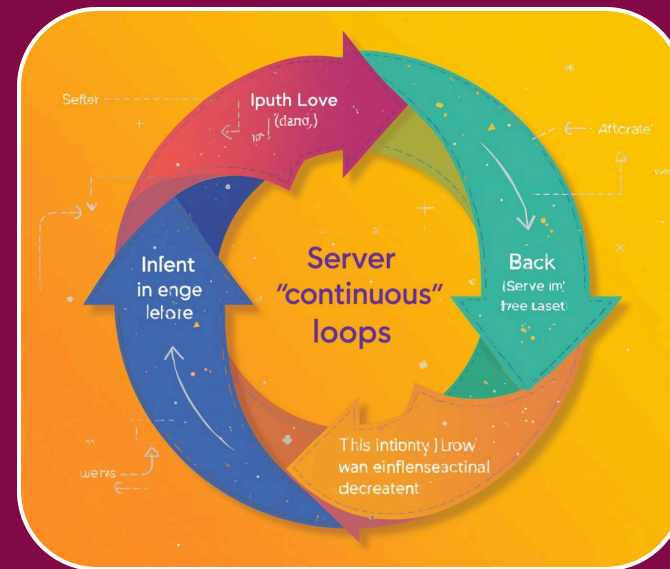
public static void main(String[] args) {
    try {
        ServidorUDP servidor = new ServidorUDP(9876);
        servidor.iniciar();
    } catch (IOException e) {
        System.err.println("✗ Error en el servidor: " + e.getMessage());
        e.printStackTrace();
    }
}
```

Puertos y Ciclo del Servidor



Puerto Fijo

El servidor utiliza un puerto fijo para recibir.



Puerto Aleatorio

El cliente usa un puerto asignado por el sistema.



Bucle Infinito

El servidor permanece en espera continua de mensajes.

Resumen Ejecutivo

La comunicación UDP permite **transferencias rápidas y eficientes** entre cliente y servidor, facilitando aplicaciones en tiempo real y dinámicas.

```
Output
ServidorUDP (run) x ServidorUDP (run) #4 x ServidorUDP (run) #5 x

===== INICIANDO CONVERSACION =====

? Enviado: Hola servidor!
? Respuesta del servidor: Servidor confirmo recepcion del mensaje: 'Hola servidor!'
-----

? Enviado: ¿Cómo estás?
? Respuesta del servidor: Servidor confirmo recepcion del mensaje: '¿Cómo estás?'
-----

? Enviado: Este es el tercer mensaje
? Respuesta del servidor: Servidor confirmo recepcion del mensaje: 'Este es el tercer mensaje'
-----

? Enviado: Enviando datos importantes
? Respuesta del servidor: Servidor confirmo recepcion del mensaje: 'Enviando datos importantes'
-----

? Enviado: Adiós servidor!
? Respuesta del servidor: Servidor confirmo recepcion del mensaje: 'Adiós servidor!'
-----

===== CONVERSACION FINALIZADA =====
```