

Intro. a Distribuidos Lab 2

Mariana Osorio Vásquez, Gabriel Jaramillo Cuberos, Roberth Méndez, Juan Esteban Vera

Agosto 2025

1 Resumen

El taller consiste en desarrollar un programa en lenguaje Java que implemente un sistema cliente-servidor multihilo. El servidor escucha en un puerto específico y, por cada cliente que se conecta, crea un nuevo hilo independiente para manejar la comunicación. Cada cliente envía un número al servidor, el cual calcula su cuadrado y devuelve el resultado al cliente. El uso de hilos permite que múltiples clientes sean atendidos de forma simultánea sin bloqueos.

1.1 Planteamiento del Problema

El problema que el código debe resolver es el siguiente:

- **Entrada:** Un número entero enviado por el cliente.
- **Objetivo:** Implementar un sistema cliente/servidor multihilo en Java donde el servidor pueda atender varios clientes simultáneamente y calcular el cuadrado de los números enviados.

El programa a desarrollar cuenta con los siguientes requisitos:

- Implementar un servidor multihilo con `ServerSocket`.
- Atender múltiples clientes de forma concurrente con hilos.
- Manejo de flujos de entrada y salida para comunicación bidireccional.
- Sincronización correcta para evitar bloqueos.
- Manejo adecuado de excepciones y cierre de recursos.

1.2 Propuesta de Solución

La solución implementa lo siguiente:

- **Servidor:**
 - Escucha en el puerto 8888 usando `ServerSocket`.

- Por cada cliente que se conecta, crea un nuevo hilo `ServerClientThread`.
- **Hilo del servidor (`ServerClientThread`):**
 - Lee el número enviado por el cliente.
 - Calcula el cuadrado del número.
 - Envía el resultado al cliente.
- **Cliente (`TCPClient`):**
 - Se conecta al servidor en el puerto 8888.
 - Envía un número leído desde consola.
 - Recibe y muestra el resultado devuelto por el servidor.

1.3 Método de Prueba

Para probar el programa, se definió el siguiente método:

1. **Prueba de conexión:**
 - Verificar que el servidor esté en escucha en el puerto 8888.
 - Ejecutar un cliente y confirmar que se conecta correctamente.
2. **Prueba de concurrencia:**
 - Ejecutar múltiples clientes simultáneamente.
 - Confirmar que todos reciben sus resultados sin bloqueos.
3. **Prueba de cálculo:**
 - Ingresar distintos números desde los clientes.
 - Verificar que el servidor devuelve el cuadrado correcto.

1.4 Resultados Esperados

Después de la ejecución del programa se espera lo siguiente:

- **Salida en el cliente:** El cuadrado del número ingresado debe ser mostrado en pantalla.
- **Salida en el servidor:** Registro de cada cliente que se conecta y el número procesado.
- **Concurrencia:** Varios clientes pueden conectarse y recibir resultados sin esperar a que otros terminen.

2 Diseño

2.1 Diseño general

- El programa está escrito en **Java**.
- El servidor utiliza:
 - **ServerSocket** para aceptar conexiones.
 - **Thread** para manejar múltiples clientes en paralelo.
 - **DataInputStream** y **DataOutputStream** para la comunicación.
- El cliente utiliza:
 - **Socket** para conectarse al servidor.
 - Entrada por consola (**BufferedReader**).
 - Flujos de datos para enviar y recibir mensajes.

2.2 Comunicación cliente-servidor

- El cliente envía un número entero en formato texto al servidor.
- El servidor procesa el número en el hilo correspondiente.
- El servidor responde con el cuadrado del número.
- La comunicación se mantiene hasta que el cliente envía el mensaje "bye".

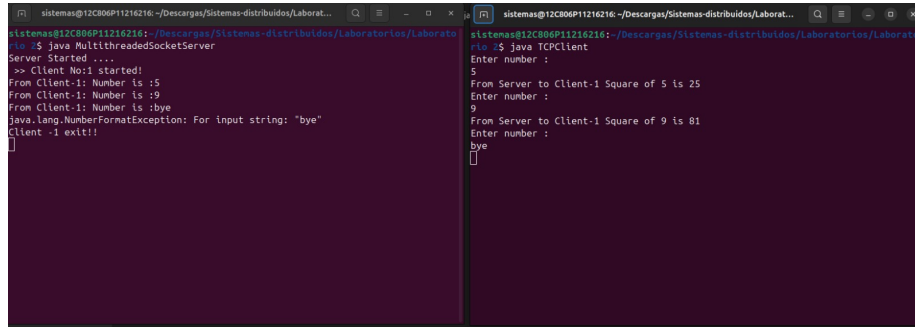
2.3 Sincronización y concurrencia

- Cada cliente es atendido en un hilo independiente.
- Los cálculos son aislados entre clientes, evitando interferencia.
- El servidor no se bloquea cuando un cliente está en espera.

2.4 Limpieza de recursos

- Los flujos de entrada y salida se cierran tras finalizar la comunicación.
- Los sockets son cerrados por ambos extremos.

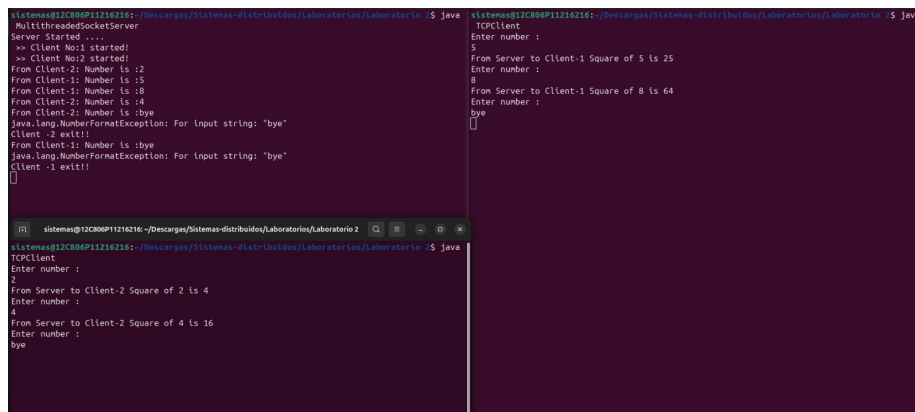
3 Resultados



```
sistemas@12C806P11216216: ~/Descargas/sistemas-distribuidos/Laborat...
$ java MultithreadedSocketServer
Server Started ....
-> Client No:1 started!
From Client-1: Number is :5
From Client-1: Number is :9
From Client-1: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -1 exit!!
]

sistemas@12C806P11216216: ~/Descargas/sistemas-distribuidos/Laborat...
$ java TOPClient
Enter number :
5
From Server to Client-1 Square of 5 is 25
Enter number :
9
From Server to Client-1 Square of 9 is 81
Enter number :
bye
]
```

Figure 1: Ejecución del servidor y un cliente local



```
sistemas@12C806P11216216: ~/Descargas/sistemas-distribuidos/Laboratorio $ java
MultithreadedSocketServer
Server Started ....
-> Client No:1 started!
-> Client No:2 started!
From Client-1: Number is :2
From Client-1: Number is :5
From Client-1: Number is :8
From Client-2: Number is :4
From Client-2: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -2 exit!!
From Client-1: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -1 exit!!
]

sistemas@12C806P11216216: ~/Descargas/sistemas-distribuidos/Laboratorio $ java
TOPClient
Enter number :
5
From Server to Client-1 Square of 5 is 25
Enter number :
8
From Server to Client-1 Square of 8 is 64
Enter number :
bye
]

sistemas@12C806P11216216: ~/Descargas/sistemas-distribuidos/Laboratorio 2 $ java
TOPClient
Enter number :
2
From Server to Client-2 Square of 2 is 4
Enter number :
4
From Server to Client-2 Square of 4 is 16
Enter number :
bye
]
```

Figure 2: Ejecución del servidor y dos clientes locales

```
sistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2 $ java
MultithreadedSocketServer
Server Started ....
>> Client No:1 started!
>> Client No:2 started!
>> Client No:3 started!
From Client-3: Number is :3
From Client-2: Number is :15
From Client-3: Number is :7
From Client-1: Number is :8
From Client-2: Number is :9
From Client-2: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -2 exits!!
From Client-1: Number is :7
From Client-3: Number is :8
From Client-3: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -3 exits!!
Client -1 exits!!

sistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2 $ java
TCPClient
Enter number :
8
From Server to Client-1 Square of 8 is 64
Enter number :
7
From Server to Client-1 Square of 7 is 49
Enter number :
bye
bye

sistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2 $ java
TCPClient
Enter number :
15
3
From Server to Client-2 Square of 15 is 225
Enter number :
9
81
From Server to Client-2 Square of 9 is 81
Enter number :
bye
bye

sistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2 $ java
TCPClient
Enter number :
3
7
From Server to Client-3 Square of 3 is 9
Enter number :
7
49
From Server to Client-3 Square of 7 is 49
Enter number :
8
64
From Server to Client-3 Square of 8 is 64
Enter number :
bye
bye
```

Figure 3: Ejecución del servidor y tres clientes locales

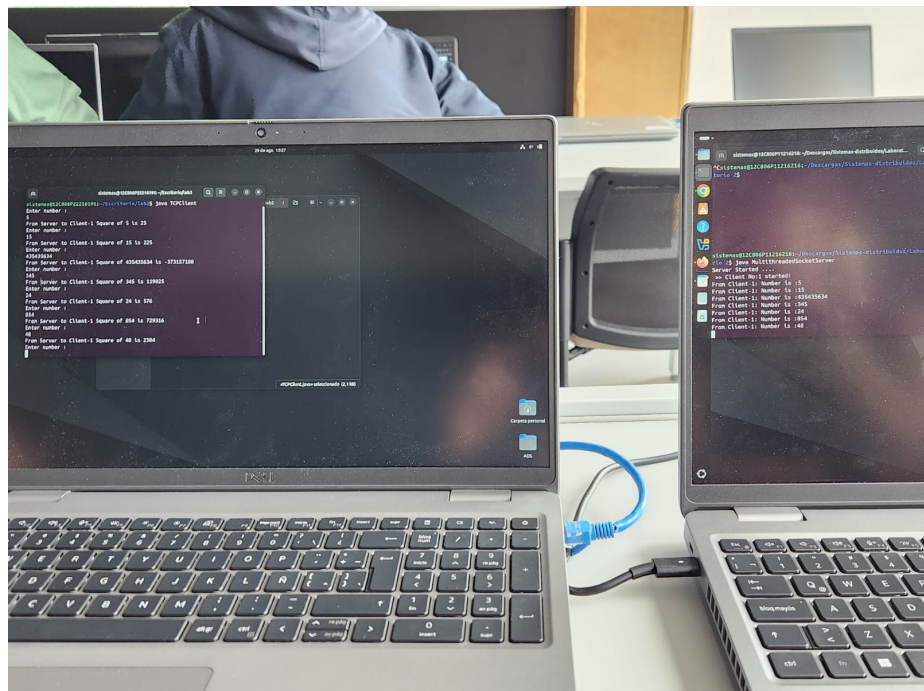


Figure 4: Ejecución del servidor y cliente remotos

```
^Csistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2$  
  
sistemas@12C806P11216216:~/Descargas/Sistemas-distribuidos/Laboratorios/Laboratorio 2$ java MultithreadedSocketServer  
Server Started ....  
>> Client No:1 started!  
From Client-1: Number is :5  
From Client-1: Number is :15  
From Client-1: Number is :435435634  
From Client-1: Number is :345  
From Client-1: Number is :24  
From Client-1: Number is :854  
From Client-1: Number is :48
```

Figure 5: Ejecución del servidor remoto

```
sistemas@12C806P22216191:~/Escritorio/lab2$ java TCPClient  
Enter number :  
5  
From Server to Client-1 Square of 5 is 25  
Enter number :  
15  
From Server to Client-1 Square of 15 is 225  
Enter number :  
435435634  
From Server to Client-1 Square of 435435634 is -373157180  
Enter number :  
345  
From Server to Client-1 Square of 345 is 119025  
Enter number :  
24  
From Server to Client-1 Square of 24 is 576  
Enter number :  
854  
From Server to Client-1 Square of 854 is 729316  
Enter number :  
48  
From Server to Client-1 Square of 48 is 2304  
Enter number :  
□
```

Figure 6: Ejecución del cliente remoto

4 Análisis de resultados

El taller demostró el funcionamiento de un sistema cliente/servidor multihilo. Se observaron los siguientes aspectos clave:

- **Manejo de múltiples clientes:** El servidor puede atender varios clientes en paralelo sin que sus interacciones se bloqueen.
- **Procesamiento concurrente:** Cada cliente recibe atención en un hilo separado, lo que mejora la eficiencia.
- **Comunicación cliente-servidor:** Se logra un intercambio bidireccional de mensajes usando flujos de datos.
- **Escalabilidad:** El diseño permite que el sistema se extienda para realizar cálculos más complejos o manejar más clientes.

5 Conclusiones

Se concluye que:

- La arquitectura multihilo en Java permite la atención concurrente de múltiples clientes en un sistema cliente-servidor.
- El uso de hilos independientes evita que un cliente bloquee a otros, garantizando un servicio eficiente.
- El manejo correcto de sockets y flujos asegura una comunicación confiable.
- La experiencia permitió comprender la importancia de la concurrencia y el paralelismo en aplicaciones distribuidas.