Taller RMI

Juan Esteban Bello Durango

Arley Bernal Muñeton

Kevin Garay Gaitan

Sistemas Distribuidos

Profesor: John Corredor Franco

1. Introducción

Este taller describe la implementación de un sistema distribuido utilizando Java RMI (Remote Method Invocation) que permite en el caso de este taller, realizar una operacion de suma entre un cliente y un servidor remoto. El sistema está compuesto por varios componentes que trabajan conjuntamente para proporcionar este servicio distribuido.

El objetivo principal de este sistema es demostrar el funcionamiento de una aplicación distribuida utilizando RMI, donde un cliente puede invocar métodos remotos en un servidor para realizar operaciones.

2. Validación de Conectividad de Red

Se validó la conectividad de red entre los dispositivos con las siguientes direcciones IP: 10.43.102.7 (servidor), 10.43.101.247 (cliente), 10.43.102.123 (cliente).

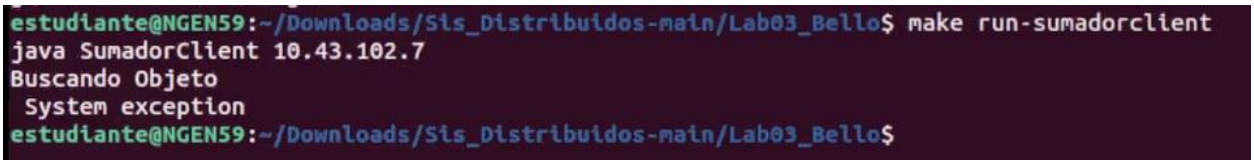
```
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.43.101.247  netmask 255.255.240.0  broadcast 10.43.111.255
        ether 00:50:56:88:63:d4  txqueuelen 1000  (Ethernet)
        RX packets 75440686  bytes 218626733805 (218.6 GB)
        RX errors 0  dropped 47179  overruns 0  frame 0
        TX packets 13659008  bytes 1976455807 (1.9 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 56238  bytes 14021300 (14.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 56238  bytes 14021300 (14.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$ ping 10.43.102.7
PING 10.43.102.7 (10.43.102.7) 56(84) bytes of data.
64 bytes from 10.43.102.7: icmp_seq=1 ttl=64 time=0.165 ms
64 bytes from 10.43.102.7: icmp_seq=2 ttl=64 time=0.171 ms
64 bytes from 10.43.102.7: icmp_seq=3 ttl=64 time=0.244 ms
64 bytes from 10.43.102.7: icmp_seq=4 ttl=64 time=0.386 ms
64 bytes from 10.43.102.7: icmp_seq=5 ttl=64 time=0.325 ms
^C
--- 10.43.102.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.165/0.258/0.386/0.086 ms
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$ ping 10.43.102.123
PING 10.43.102.123 (10.43.102.123) 56(84) bytes of data.
64 bytes from 10.43.102.123: icmp_seq=1 ttl=64 time=3.86 ms
64 bytes from 10.43.102.123: icmp_seq=2 ttl=64 time=0.530 ms
64 bytes from 10.43.102.123: icmp_seq=3 ttl=64 time=0.446 ms
64 bytes from 10.43.102.123: icmp_seq=4 ttl=64 time=0.437 ms
64 bytes from 10.43.102.123: icmp_seq=5 ttl=64 time=0.561 ms
^C
--- 10.43.102.123 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4090ms
rtt min/avg/max/mdev = 0.437/1.166/3.856/1.345 ms
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$
```

Mediante el comando ping, se confirmó que existe conectividad entre estos dispositivos, lo que es fundamental para el correcto funcionamiento del sistema RMI implementado. Todos los dispositivos respondieron satisfactoriamente a las solicitudes de ping, confirmando que la infraestructura de red está correctamente configurada para permitir la comunicación entre ellos. Esto se consiguió utilizando las maquinas remotas proveidas por la universidad con sistema operativo Ubuntu, por lo que todos los dispositivos se encontraban en la misma red local

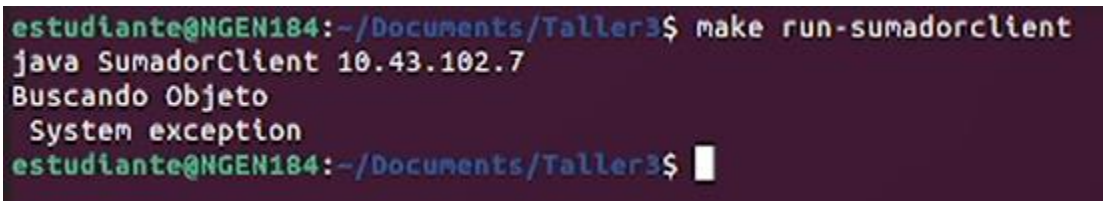
3. Casos de Prueba

Caso 1: Se ejecuta el cliente sin haber iniciado previamente el servidor.

Cliente 1 (10.43.101.247):



Cliente 2 (10.43.102.123):



Resultado: El cliente muestra el mensaje "Buscando Objeto" pero no puede encontrar el objeto remoto, resultando en un error de conexión.

Análisis: El cliente intenta localizar el objeto remoto, pero al no estar ejecutándose el servidor, el registro RMI no tiene el objeto referenciado lo que genera una excepción "System exception"

Caso 2: Ejecución normal del servidor y cliente

Se inicia el servidor (10.43.102.7) con make run-sumadorserver

```
estudiante@NGEN72:~/Downloads/programa$ make run-sumadorserver
java -classpath . -Djava.security.policy=./policy SumadorServer
Rebind objeto MiSumador
```

Se ejecuta el cliente con make run-sumadorclient proveyendo la ip del servidor en el makefile

Cliente 1 (10.43.101.247):

```
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$ make run-sumadorclient
java SumadorClient 10.43.102.7
Buscando Objeto
5 + 2 = 7
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$
```

Cliente 2 (10.43.102.123):

```
estudiante@NGEN184:~/Documents/Taller3$ make run-sumadorclient
java SumadorClient 10.43.102.7
Buscando Objeto
5 + 2 = 7
estudiante@NGEN184:~/Documents/Taller3$
```

Resultado: El sistema calcula correctamente 5 + 2 = 7

Análisis: El servidor se inicia correctamente, crea una instancia de SumadorImpl y la registra en el RMI registry con el nombre "MiSumador", el cliente localiza el objeto remoto y invoca el método sumar(a, b), l método remoto ejecuta la operación y devuelve el resultado al cliente

Caso 3: Cambio de valores y nueva ejecución

Se modifica el código del cliente (SumadorCliente.java) para cambiar los valores de 5 y 2 a 8 y 1

```java
// Cliente para el servicio Sumador
public class SumadorClient {
    public static void main(String args[]) {
        int res = 0;
        // Numeros a sumar
        int a = 5;
        int b = 2;
        try {
            System.out.println("Buscando Objeto ");
```

```java
// Cliente para el servicio Sumador
public class SumadorClient {
    public static void main(String args[]) {
        int res = 0;
        // Numeros a sumar
        int a = 1;
        int b = 8;
        try {
            System.out.println("Buscando Objeto ");
```

Se compila y ejecuta nuevamente el cliente (con el servidor aún en ejecución)

```
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$ make run-sumadorclient
java SumadorClient 10.43.102.7
Buscando Objeto
1 + 8 = 9
estudiante@NGEN59:~/Downloads/Sis_Distribuidos-main/Lab03_Bello$
```

Resultado: El sistema calcula correctamente 8 + 1 = 9

Análisis: La modificación de los valores en el código cliente no afecta al servidor

Estructura del Sistema

- Makefile: Contiene las instrucciones para compilar y ejecutar el sistema
- policy: Define los permisos de seguridad para la ejecución RMI
- Sumador.java: Interfaz que define los métodos remotos disponibles
- SumadorImpl.java: Implementación de la interfaz remota

- SumadorServer.java: Servidor que publica el objeto remoto
- SumadorClient.java: Cliente que consume el servicio remoto

4. Conclusión

El sistema implementado demuestra correctamente la implementación de sistemas distribuidos utilizando RMI. Los casos de prueba validan el comportamiento esperado en diferentes escenarios, confirmando funcionalidad del sistema yla capacidad de modificar el cliente sin afectar al servidor, lo que evidencia el desacoplamiento característico de los sistemas distribuidos, donde los componentes pueden evolucionar de manera independiente mientras mantienen una interfaz de comunicación bien definida.