

## Taller 1 - Principios de Sistemas Distribuidos



Pontificia Universidad  
**JAVERIANA**  
Colombia

Maria Paula Rodríguez Ruiz

Daniel Felipe Castro Moreno

Juan Enrique Rozo Tarache

Eliana Katherine Cepeda González

Introducción a Sistemas Distribuidos

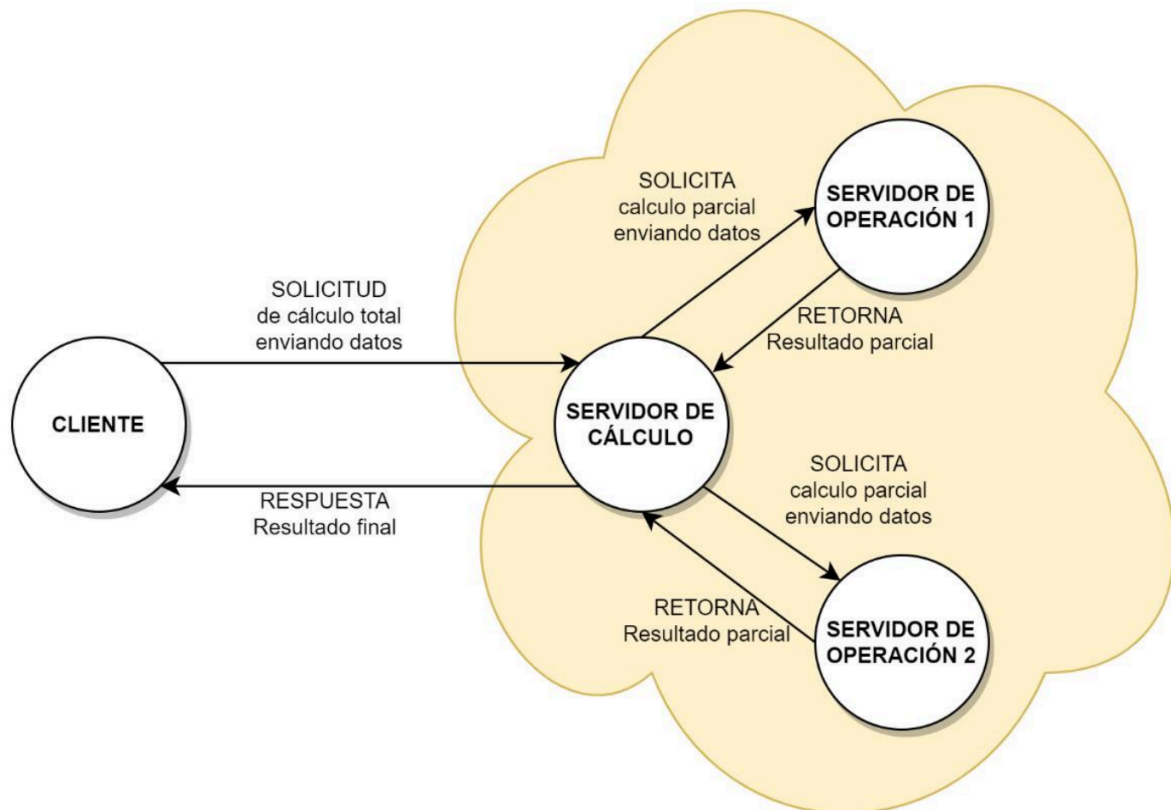
Dpto. de Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá, Colombia

16 de Febrero del 2025

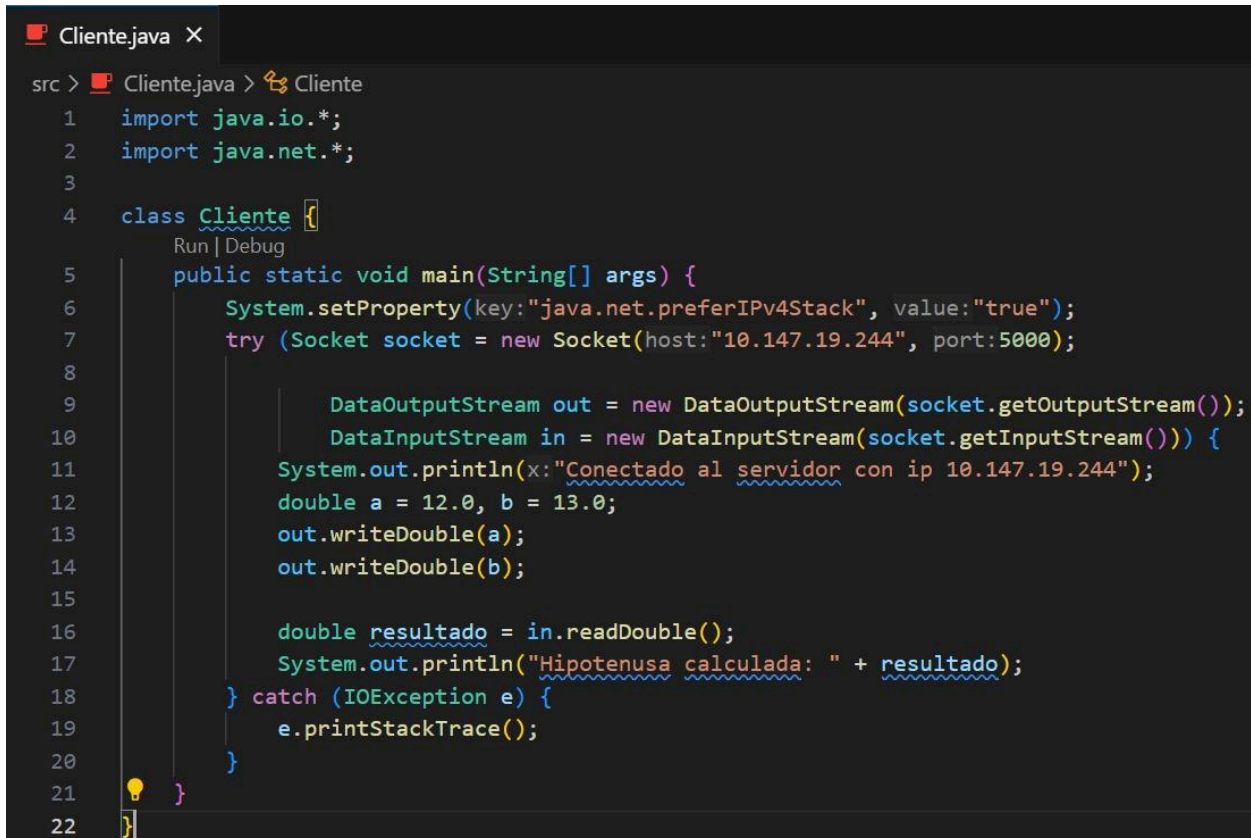
Para este taller se implementó una aplicación de cálculo distribuido usando sockets como medio de comunicación sin hacer uso de algún middleware. La arquitectura básica se muestra en la Figura 1 a continuación.



*Figura 1. Arquitectura básica*

Su funcionamiento es el siguiente:

1. **Cliente:** Se ejecuta en su propia máquina y envía los valores A y B al **servidor de cálculo**. Para luego, recibir el resultado final. Su implementación en Java es la siguiente:



```
src > Cliente.java > Cliente
1  import java.io.*;
2  import java.net.*;
3
4  class Cliente {
5      public static void main(String[] args) {
6          System.setProperty(key:"java.net.preferIPv4Stack", value:"true");
7          try (Socket socket = new Socket(host:"10.147.19.244", port:5000);
8
9              DataOutputStream out = new DataOutputStream(socket.getOutputStream());
10             DataInputStream in = new DataInputStream(socket.getInputStream())) {
11              System.out.println(x:"Conectado al servidor con ip 10.147.19.244");
12              double a = 12.0, b = 13.0;
13              out.writeDouble(a);
14              out.writeDouble(b);
15
16              double resultado = in.readDouble();
17              System.out.println("Hipotenusa calculada: " + resultado);
18          } catch (IOException e) {
19              e.printStackTrace();
20          }
21      }
22  }
```

Figura 2. Programa del Cliente en máquina con IP: 10.147.19.109

2. **Servidor de Cálculo:** Se ejecuta en otra máquina y:
  - Recibe los valores del **cliente**.
  - Envía A y B hacia un **servidor de operación** respectivamente.
  - Si un **servidor de operación** no responde, calcula el cuadrado de ese número localmente.
  - Suma los resultados y calcula la raíz cuadrada.

- Envía el resultado al **cliente**.

Su implementación en Java es la siguiente:

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

class ServidorCalculo {
    Run | Debug
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(port:5000, backlog:50, InetAddress.getByName(host:"10.147.19.244"))) {
            System.out.println("Servidor de cálculo esperando cliente...");
            Socket clienteSocket = serverSocket.accept();
            System.out.println("Cliente conectado con direccion: " + clienteSocket.getInetAddress().getHostAddress());

            DataInputStream in = new DataInputStream(clienteSocket.getInputStream());
            DataOutputStream out = new DataOutputStream(clienteSocket.getOutputStream());

            double a = in.readDouble();
            double b = in.readDouble();

            // Se envían valores al servidor de operación
            double a2 = enviarOperacion(a, ip:"10.147.19.224", puerto:5001); // Servidor de Operación
            double b2 = enviarOperacion(b, ip:"10.147.19.5", puerto:5001); // Servidor de Operación

            double hipotenusa = Math.sqrt(a2 + b2);
            System.out.println("Enviando resultado: " + hipotenusa);
            out.writeDouble(hipotenusa);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static double enviarOperacion(double valor, String ip, int puerto) {
        try (Socket socket = new Socket(ip, puerto);
            DataOutputStream out = new DataOutputStream(socket.getOutputStream());
            DataInputStream in = new DataInputStream(socket.getInputStream())) {
            out.writeDouble(valor);
            System.out.println("Servidor en " + ip + " respondió correctamente.");

            double resultado = in.readDouble(); // Guardamos el resultado en una variable
            System.out.println("Resultado calculado remotamente: " + resultado);

            return resultado; // Retornamos la misma variable
        } catch (IOException e) {
            System.out.println("Servidor en " + ip + " no responde. Calculando localmente.");
            System.out.println("Resultado calculado localmente: " + (valor * valor));
            return valor * valor;
        }
    }
}
```

*Figura 3. Programa del Servidor de Cálculo en máquina con IP: 10.147.19.244*

3. **Servidores de Operación (2 máquinas diferentes):** Cada uno recibe un número, lo eleva al cuadrado y lo devuelve. La implementación en Java en la máquina 1 es la siguiente:

```
1 import java.io.*;
2 import java.net.*;
3
4 class ServidorOperacion {
5     public static void main(String[] args) {
6         try (ServerSocket serverSocket = new ServerSocket(port: 5001, backlog: 50, InetAddress.getByName(host: "10.147.19.224"))) {
7             System.out.println("Servidor de operación escuchando en puerto 5001");
8             while (true) {
9                 Socket socket = serverSocket.accept();
10                System.out.println(" Cliente conectado desde: " + socket.getInetAddress().getHostAddress());
11                new Thread(new ManejadorOperacion(socket)).start();
12            }
13        } catch (IOException e) {
14            e.printStackTrace();
15        }
16    }
17 }
18
```

Figura 4. Primera parte del programa del Servidor de Operación 1 en máquina con IP: 10.147.19.224

```
19 class ManejadorOperacion implements Runnable {
20     4 usages
21     private Socket socket;
22
23     1 usage
24     public ManejadorOperacion(Socket socket) { this.socket = socket; }
25
26     @Override
27     public void run() {
28         try (DataInputStream in = new DataInputStream(socket.getInputStream());
29              DataOutputStream out = new DataOutputStream(socket.getOutputStream())) {
30
31             double valor = in.readDouble();
32             System.out.println(" Recibi el valor: " + valor + " de " + socket.getInetAddress().getHostAddress());
33
34             double resultado = valor * valor;
35             System.out.println("Enviando resultado: " + resultado);
36
37             out.writeDouble(resultado);
38         } catch (IOException e) {
39             e.printStackTrace();
40         }
41     }
42 }
```

Figura 5. Segunda parte del programa del Servidor de Operación 1 en máquina con IP: 10.147.19.224

La implementación en Java en la máquina 2 es la siguiente:

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

class ServidorOperacion {
    Run | Debug
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(port:5001, backlog:50, InetAddress.getByName(host:"10.147.19.5"))) {
            System.out.println(x:"Servidor de operación escuchando en puerto 5001");
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("Servidor de Cálculo conectado desde: " + socket.getInetAddress().getHostAddress());
                new Thread(new ManejadorOperacion(socket)).start();
                return;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figura 6. Primera parte del programa del Servidor de Operación 2 en máquina con IP: 10.147.19.5

```
class ManejadorOperacion implements Runnable {
    private Socket socket;

    public ManejadorOperacion(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        try (DataInputStream in = new DataInputStream(socket.getInputStream());
            DataOutputStream out = new DataOutputStream(socket.getOutputStream())) {

            double valor = in.readDouble();
            System.out.println(" Recibí el valor: " + valor);

            double resultado = valor * valor;
            System.out.println("Enviando resultado: " + resultado);

            out.writeDouble(resultado);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figura 7. Segunda parte del programa del Servidor de Operación 2 en máquina con IP: 10.147.19.5

Para el funcionamiento de la anterior implementación se hizo uso de ZeroTier, una aplicación que permite crear una VPN privada. Esta fue instalada en las cuatro máquinas, a las cuales se les asignó una IP para establecer la conexión a la red privada virtual. Luego, se modificó el programa de los servidores y del cliente para comunicarse utilizando las nuevas direcciones IP asignadas por la VPN como se muestra en la figura 8 en lugar de las de red local, permitiendo la interacción entre las computadoras independientemente de la ubicación.

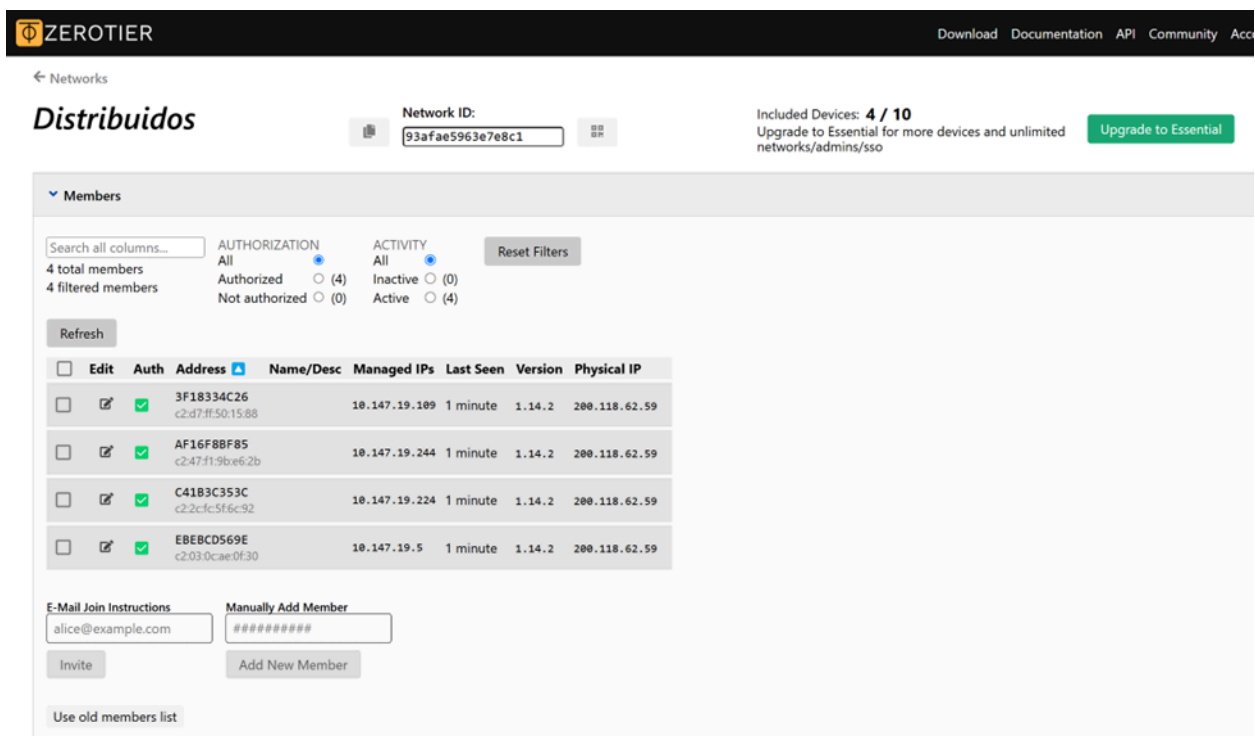


Figura 8. Computadoras conectadas a la red privada virtual

La prueba de funcionamiento se presenta a continuación. Primero, el servidor inicia la comunicación, escuchando en el puerto 5000 al cliente. Una vez conectado, el cliente publica los valores de los catetos ( $A=12$  y  $B=13$ ) pertenecientes a un triángulo rectángulo y se ejecuta el programa con los servidores de operación 1 (10.144.19.224) y 2 (10.144.19.5) en el orden respectivo. En otras palabras, el Servidor de Operación 1 recibe el cateto de longitud 12, calcula su cuadrado (144) y lo envía al servidor principal, y



solamente después el Servidor de Operación 2 hace lo mismo con el cateto de longitud 13 (169).

Finalmente, el servidor suma los cuadrados y obtiene su raíz, lo cual corresponde al valor de la longitud de la hipotenusa que envía al cliente, finalizando así su ejecución y la del cliente. Los operadores lo hacen al terminar sus cálculos. A continuación, se presentan las distintas ejecuciones:

```
PS C:\Users\thega\OneDrive\Documentos\VisualJava\ServidorCalculo> cd;
2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Use
Servidor de cálculo esperando cliente...
Cliente conectado con direccion: 10.147.19.109
Servidor en 10.147.19.224 respondió correctamente.
Resultado calculado remotamente: 144.0
Servidor en 10.147.19.5 respondió correctamente.
Resultado calculado remotamente: 169.0
Enviando resultado: 17.69180601295413
PS C:\Users\thega\OneDrive\Documentos\VisualJava\ServidorCalculo>
```

Figura 9. Ejecución del Servidor de Cálculo.

```
Servidor de operación escuchando en puerto 5001
Servidor de Cálculo conectado desde: 10.147.19.244
Recibí el valor: 12.0
Enviando resultado: 144.0

Process finished with exit code 0
```

Figura 10. Ejecución del Servidor de Operación 1.

```
PS C:\Users\ana.moreno\Downloads\Distri\Distri> cd;
etailsInExceptionMessages' '-cp' 'C:\Users\ana.moreno\Downloads\Distri\Distri
Servidor de operación escuchando en puerto 5001
Servidor de Cálculo conectado desde: 10.147.19.244
Recibí el valor: 13.0
Enviando resultado: 169.0
PS C:\Users\ana.moreno\Downloads\Distri\Distri>
```

Figura 11. Ejecución del Servidor de Operación 2.



```
Conectado al servidor con ip 10.147.19.244
Hipotenusa calculada: 17.69180601295413
```

Figura 12. Ejecución del cliente.

El comportamiento presentado anteriormente se describe en el siguiente diagrama:

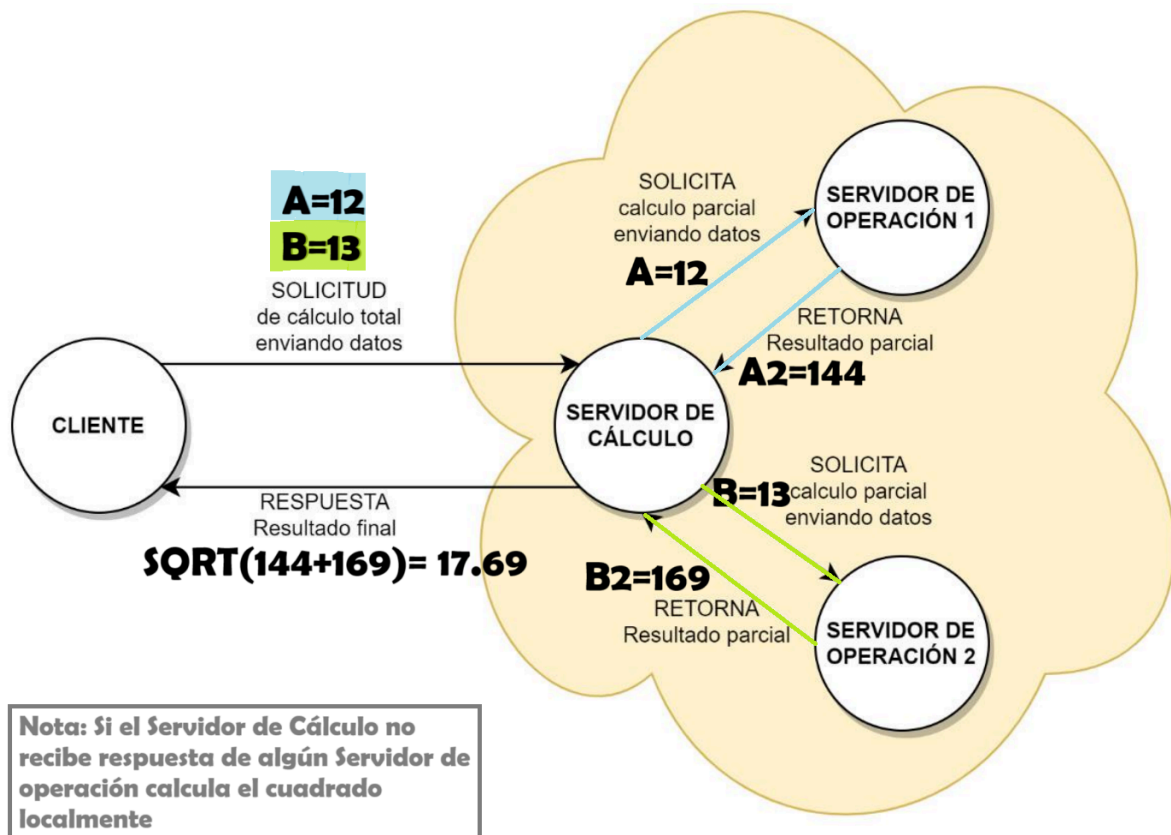


Figura 13. Arquitectura de la prueba de funcionamiento.