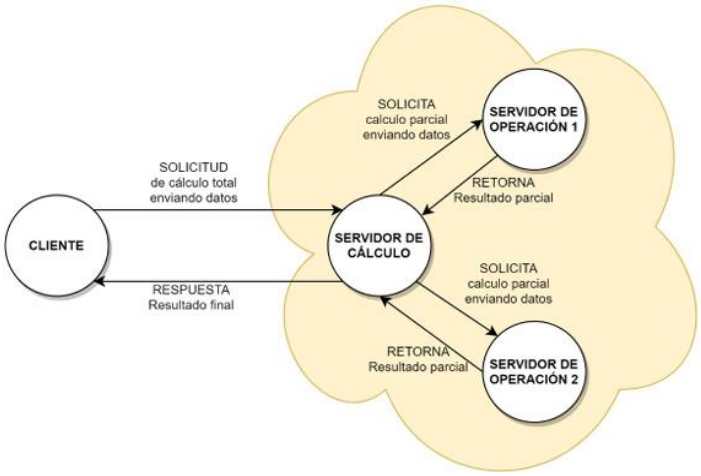
Taller 1
Sistemas Distribuidos

Profesor: Osberth Cristhian Luef De Castro Cuevas
Estudiantes:
Julián Andres Arana Guiza,
Angie Valentina León González,
Isabella Blanco Chaparro,
Juan David Barajas Urrea.

Explicación del desarrollo del taller 1:



Grafica 1: Topología de la conexión

Ejecución:

- Se ejecutan primero de forma individual los servidores 1 y 2, después el servidor principal y el ultimo por ejecutar es el cliente quien es el que envía la lista de números.

CLIENTE:

- Se declara una variable final Host con el valor "127.0.0.1", que representa la dirección IP del servidor al que se conectará el cliente.
- Se declara una variable final puerto con el valor 5000, que representa el puerto en el servidor al que se conectará el cliente.
- Se declara un array arreglo de enteros con algunos valores.
- Se inicializan los objetos in y out de tipo DataInputStream y DataOutputStream, respectivamente, para la entrada y salida de datos desde y hacia el socket.
- Se crea un nuevo objeto Socket para establecer una conexión con el servidor utilizando la dirección IP y el puerto especificado.
- Se envía la longitud del array arreglo al servidor utilizando el método writeInt del objeto out.
- Se envían los elementos del array arreglo uno por uno al servidor utilizando el bucle for.
- Se lee un entero del servidor utilizando el método readInt del objeto in y se almacena en la variable mensaje.
- Se imprime el mensaje recibido del servidor.
- Se cierra el socket después de completar las operaciones.

SERVIDOR:

- Se declara una variable servidor de tipo ServerSocket inicializada en null.
- Se definen constantes para los puertos y las direcciones IP de los servidores secundarios.
- Se crea un nuevo objeto ServerSocket en el puerto especificado y se imprime un mensaje indicando que el servidor ha iniciado.
- Se entra en un bucle infinito para esperar conexiones entrantes de clientes.
- Cuando se establece una conexión con un cliente, se imprime un mensaje indicando que el cliente se ha conectado.
- Se crean objetos DataInputStream y DataOutputStream para la entrada y salida de datos desde y hacia el cliente, respectivamente.
- Se lee la longitud del arreglo de enteros enviado por el cliente y se crea un arreglo para almacenar los valores.
- Se divide el arreglo en dos mitades.
- Se envían las mitades del arreglo a diferentes servidores secundarios para su procesamiento utilizando el método enviarYRecibirMensaje.
- Se suma las respuestas recibidas de los servidores secundarios.
- Se envía la suma al cliente utilizando el objeto out.
- Se cierra el socket después de completar las operaciones y se imprime un mensaje indicando que el cliente se ha desconectado.
- Metodo 'EnviaryRecibirmensajes':
    - Este método se utiliza para enviar un arreglo de enteros a un servidor secundario y recibir un resultado.
    - Se crea un nuevo socket para conectarse al servidor secundario.
    - Se crean objetos DataInputStream y DataOutputStream para la entrada y salida de datos desde y hacia el servidor secundario, respectivamente.
    - Se envía la longitud del arreglo de enteros y los valores del arreglo al servidor secundario.
    - Se lee y devuelve la respuesta del servidor secundario.

SERVIDOR 1:

- Se declara una variable servidor de tipo ServerSocket inicializada en null.
- Se declara una variable sc de tipo Socket inicializada en null.
- Se define una constante puerto que representa el puerto en el que el servidor escucha conexiones entrantes.
- Se declara un DataInputStream llamado in y un DataOutputStream llamado out.
- Se crea un nuevo objeto ServerSocket en el puerto especificado y se imprime un mensaje indicando que el servidor ha iniciado.
- Se entra en un bucle infinito para esperar conexiones entrantes de clientes.
- Cuando se establece una conexión con un cliente, se imprime un mensaje indicando que el servidor principal se ha conectado.
- Se crean objetos DataInputStream y DataOutputStream para la entrada y salida de datos desde y hacia el cliente, respectivamente.
- Se lee la longitud del arreglo de enteros enviado por el cliente y se crea un arreglo para almacenar los valores.
- Se lee cada elemento del arreglo y se calcula la suma de todos los elementos.
- Se envía la suma al cliente utilizando el objeto out.

- Se cierra el socket después de completar las operaciones y se imprime un mensaje indicando que el servidor principal se ha desconectado.

SERVIDOR 2:

- Se declara una variable servidor de tipo ServerSocket inicializada en null.
- Se declara una variable sc de tipo Socket inicializada en null.
- Se define una constante puerto que representa el puerto en el que el servidor escucha conexiones entrantes.
- Se declara un DataInputStream llamado in y un DataOutputStream llamado out.
- Se crea un nuevo objeto ServerSocket en el puerto especificado y se imprime un mensaje indicando que el servidor ha iniciado.
- Se entra en un bucle infinito para esperar conexiones entrantes de clientes.
- Cuando se establece una conexión con un cliente, se imprime un mensaje indicando que el servidor principal se ha conectado.
- Se crean objetos DataInputStream y DataOutputStream para la entrada y salida de datos desde y hacia el cliente, respectivamente.
- Se lee la longitud del arreglo de enteros enviado por el cliente y se crea un arreglo para almacenar los valores.
- Se lee cada elemento del arreglo y se calcula la suma de todos los elementos.
- Se envía la suma al cliente utilizando el objeto out.
- Se cierra el socket después de completar las operaciones y se imprime un mensaje indicando que el servidor principal se ha desconectado.