Informe Trabajo Practico N°1

Conceptos básicos para la construcción de Sistemas Distribuidos

Alumno: Garrós Gastón
Legajo: 118286
Materia: Sistemas Distribuidos y Programación Paralela

1. Escriba un servidor que, usando sockets, reciba un mensaje de texto y lo repita a su cliente. Programe también el cliente para verificar y probar el comportamiento del servidor. Realice la implementación en TCP y comente los resultados.

   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:
   I. Server.java
   II. Cliente.java

   Una vez aceptada la conexión en el servidor este bloquea ese par ip:puerto hasta el cliente termine de forma correcta la conversación.
   Este tipo de implementación solo puede aceptar un cliente a la vez.

2. Modifique el programa anterior para que pueda atender varios clientes a la vez.

   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:
   I. Server.java
   II. Cliente.java

   Este tipo de implementación puede aceptar varios clientes a la vez, esto lo hace teniendo un único puerto para que el cliente envié peticiones de conexión pero el server para poder aceptarlas crea thread que sirven servicios en diferentes puertos y se conecta a estos puertos. Esto para el usuario es transparente y es una gran mejora para la implementación anterior ya que permite tener mayor disponibilidad para usuarios porque el server no se bloquea con una sola conexión.

3. Escriba un servidor de mensajes en colas, que permita a los clientes dejar un mensaje (identificando de alguna forma a quién se lo dejan), y bajar los mensajes que le están dirigidos. La comunicación entre cliente y servidor debe ser mediante sockets, y el servidor debe poder atender varios clientes a la vez.
   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:
   I. Server.java
   II. Cliente.java

   El servidor atiende varios clientes utilizando thread y utiliza una cola de mensajes para tener un registro de mensajes para cada uno de los clientes que se conecte, esta cola es volcada a disco para posteriores ingresos, este volcado se realiza en formato json.

4. Modifique el programa anterior para que el mensaje de la cola sea borrado por el servidor una vez que el cliente confirma, mediante un mensaje de tipo ack, que efectivamente recibió el mensaje que estaba en la cola.
   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:
   I. Server.java
   II. Cliente.java

   El servidor atiende varios clientes utilizando thread y utiliza una cola de mensajes para tener un registro de mensajes para cada uno de los clientes que se conecte, esta cola es volcada a disco para posteriores ingresos, este volcado se realiza en formato json.
   El servidor para eliminar el mensajes lee el archivo elimina el mensajes y lo graba en disco actualizado.

5. Escribir un servicio que devuelva información de clima del lugar donde reside el servidor. Esta información podrá generarse de forma aleatoria. Deberá ser realizado con RMI. Para ello considere la interface remota, la clase (lado servidor) que implementa esa interface, el servidor, y un cliente que permita probar este funcionamiento.
   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:

I. ServerRmi.java
II. Cliente.java

El cliente se conecta con el servidor RMI el cual ofrece una lista de servicio, el cliente selecciona el servicio deseado en este caso “InfoClima” y realiza consultas a través de la interfaz correspondientemente, sobre ese servicio que se ejecutará en el servidor. Varios clientes pueden ejecutarse al mismo tiempo consultando por el mismo servicio sin que este sea bloqueado.

6. Escribir un servidor utilizando RMI, que ofrezca la posibilidad de sumar y restar vectores de enteros. Introduzca un error en su código que modifique los vectores recibidos por parámetro. Qué impacto se genera? Qué conclusión saca sobre la forma de pasaje de parámetros en RMI? Mostrar claramente los valores de los vectores del lado cliente, antes y después de la ejecución de la suma o resta.

   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:

   I. ServerRmi.java
   II. Cliente.java

   El cliente se conecta con el servidor RMI el cual ofrece una lista de servicio, el cliente selecciona el servicio deseado en este caso “OperacionesVectores” y realiza consultas a
   III.
   IV. través de la interfaz correspondiente, sobre ese servicio que se ejecutará en el servidor.

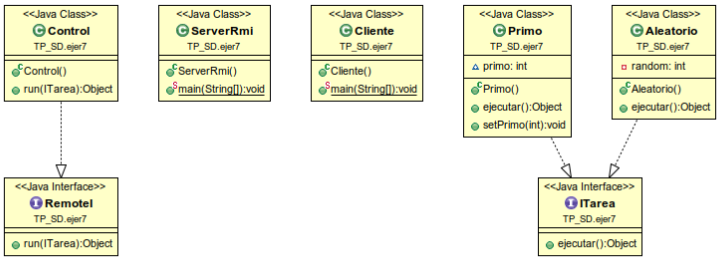
   Un error fue introducido del lado del servidor para evaluar que impacto tenia sobre los parametros enviados por el cliente y el resultado fue que como los parámetros son pasados por valor en el cliente no llegan cambios. El error introducido fue colocar en el V1 un vector nulo, pero luego cuando el cliente imprime los datos de los vectores son los mismo ingresados. Por lo tanto se llego a la conclusión que si se desea ver lo cambios se deben pasar los cambios al cliente y este deberá imprimirlos sabiendo sobre estos cambios.

7. Implemente un servidor RMI que resuelva tareas genéricas. Para ello tener en cuenta la interface Tarea, que tendrá un método ejecutar(). El objetivo es que desde el cliente se puedan escribir objetos (que implementen la interface Tarea) que hagan un cálculo concreto (calcular un número aleatorio, un primo, el valor de Pi con cierta precisión, etc.), y que esa tarea se pase al servidor RMI, quien hará el cálculo y devolverá el valor al cliente.|

   La implementación se realizo en java sin parámetros, debe ejecutarse de la siguiente forma:

   I. ServerRmi.java
   II. Cliente.java

   El cliente se conecta con el servidor RMI el cual ofrece una lista de servicio, el cliente selecciona el servicio deseado en este caso “OperacionesMatematicas” y realiza consultas a través de la interfaz correspondiente, sobre ese servicio que se ejecutará en el servidor.



La clase control implementa la Interfaz RemoteI al igual que Primo y Aleatorio implementan Itarea. Entonces el Cliente se comunica con el ServerRmi a traves de la interfaz RemoteI pasando

las instancias de las clases creadas en el cliente. El servidor hace el bind con la clase Control que es la que implementa el metodo de la interfaz RemotoI y ejecuta a través de la interfaz ITarea el metodo correspondiente a cada una de las clases Primo y Aleatorio, devolviendo el resultado al cliente. Para que las instancias enviadas por el cliente lleguen al servidor y vuelvan sin crear copias deben implementar Serializable.