Universidad Nacional
ARTURO JAURETCHE

Sistemas Distribuidos

Trabajo Práctico n°3

Alumna: Cabral Sabrina Lourdes

El objetivo del laboratorio fue familiarizar al estudiante con el desarrollo de aplicaciones distribuidas basadas en CORBA. Además de introducir los conceptos de llamadas a procedimientos remotos desde las aplicaciones.

Para el desarrollo del Laboratorio, en una primera instancia se empleó la imagen proporcionada por la catedra (distribución de Debian 10) pero, para la misma no estaban disponibles los paquetes orbit e idl (necesarios para el desarrollo del laboratorio). Es por ello, que se utilizó la distribución de Debian 9. Luego de virtualizar la imagen del sistema operativo en Virtual Box, y configurar la carpeta compartida, se procedió a instalar los siguientes paquetes libidl-2-0, orbit2 y default-jdk, para el último paquete, primero se verifico que java ya se encuentre instalado, en caso de no estarlo debía ser instalado.

```
root@Servidor:/home/sabrina# java -version
openjdk version "1.8.0_252"
OpenJDK Runtime Environment (build 1.8.0_252-8u252-b09-1~deb9u1-b09)
OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)
root@Servidor:/home/sabrina#
```

Una vez instalados los paquetes anteriormente mencionados, se ejecutó el programa de ejemplo proporcionado por la catedra. El mismo, está compuesto por tres archivos, uno .idl y dos .java (uno es el cliente y el otro es el servidor).

```
root@Debian:/home# cd /media/sf_CORBA/Laboratorio\ 3/
root@Debian:/media/sf_CORBA/Laboratorio 3# ls
Cliente.java  holamundo.idl  Servidor.java
root@Debian:/media/sf_CORBA/Laboratorio 3#
```

En primero lugar se debe ejecutar el archivo .idl, en este se definen las interfaces entre los componentes de la aplicación, y es el que asegura la independencia del lenguaje, ya que no en él no se realiza la implementación de cada método, sino que solo se los define.

```
root@Debian:/media/sf_CORBA/Laboratorio 3# idlj -fall holamundo.idl
root@Debian:/media/sf_CORBA/Laboratorio 3#
```

El compilador se encargará de generar los archivos que conectan los métodos definidos en la interfaz, con su respectiva implementación.

Luego se debe compilar todos los archivos, tanto los que se encuentren en el directorio donde se está trabajando, como los archivos generados en el paso anterior.

```
root@Debian:/media/sf_CORBA/Laboratorio 3# javac -d hola *.java hola/*.java
Note: hola/holamundoPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
root@Debian:/media/sf_CORBA/Laboratorio 3#
```

Para iniciar el servidor se debe utilizar el siguiente comando.

```
root@Debian:/media/sf_CORBA/Laboratorio 3# java -cp .:hola hola.Servidor
```

En otro terminal, (de la misma maquina) se de ejecutar el cliente, que se encargará de invocar el saludo, implementado en el servidor.

```
root@Debian:/home/sabrina# cd /media/sf_CORBA/Laboratorio\ 3/
root@Debian:/media/sf_CORBA/Laboratorio 3# java -cp .:hola hola.Cliente
root@Debian:/media/sf_CORBA/Laboratorio 3#
```

En la terminal que se está ejecutando el servidor se podrá visualizar la respuesta de la llamada invocada por el cliente.

```
root@Debian:/media/sf_CORBA/Laboratorio 3# java -cp .:hola hola.Servidor
Hola Mundo!!
```

Desarrollo del codigo para la implementacion de calculadora con CORBA

En pimer lugar, se definió el archivo calculo.idl, en el se establecen los metodos que luego seran implementados por el servidor. Para la calculadora se definieron cuatro metodos: suma, resta, multiplicacion y division.

```
module calculadora {
	interface calculo {
	long suma (in long a, in long b);
	long resta (in long a, in long b);
	long multiplicacion (in long a, in long b);
	double division (in long a, in long b);
	};
};
```

Luego, se procedio al desarrollo de la clase cliente, en ella se importó el modulo que contiene la definicion de la interfaz, ademas de la librería org.omg.CosNaming que permitira utilizar el servicio de nombres del ORB, java.util.Scanner que permite ingresar texto por consola, org.omg.CORBA que es necesario para hacer uso de las funciones de CORBA.

```
//
// Cliente.java
//
package calculadora;

import org.omg.CORBA.ORB;
import java.util.Scanner;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

import java.io.*;
```

Luego se definio la clase Cliente, en ella se inicia el ORB y se emplea el servicio de nombres para que el cliente pueda ubicar los servicios que proporciona el Servidor, pero para poder emplearlo, primero se debe obtener el contexto inicial.

```
public class Cliente {
public static void main (String args[ ]) {

    try {
        // INICIAMOS ORB
        ORB orb = ORB.init(args, null);
        org.omg.CORBA.Object objRef =  orb.resolve_initial_references("NameService");
        NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
        String name = "Calculo";
```

Luego se obtiene la referencia al objeto, es decir se obtiene la referencia al servidor, y con ello se puede acceder a los metodos definidos en él.

```
calculo cuentas = calculoHelper.narrow(ncRef.resolve_str(name));

Scanner scan = new Scanner(System.in);
int opcion;
int dato1;
int dato2;

System.out.println("Ingrese un numero: ");
dato1 = scan.nextInt();

System.out.println("Ingrese un numero: ");
dato2 = scan.nextInt();

System.out.println("Ingrese una opcion: ");
System.out.println("Ingrese 1 para la suma: ");
System.out.println("Ingrese 2 para la resta: ");
System.out.println("Ingrese 3 para la multiplicacion: ");
System.out.println("Ingrese 4 para la division: ");

opcion =scan.nextInt();
```

Posterimente, se desarrolla el codigo que va a solcitar al cliente que ingrese dos numero por teclado, cada uno es almacenado en una variable diferente, y luego de ingresar los datos, se desplegara un menu donde se debe ingresar el tipo de operación a realizar.

Dependiendo del tipo de operación se realizará la llamada al metodo que se encuentra implementado en el servidor.

```
    opcion =scan.nextInt();
    if (opcion==1){
        System.out.println("el resultad de la suma es : "+cuentas.suma(dato1,dato2));
    }
    if (opcion==2){
        System.out.println("el resultad de la resta es : "+ cuentas.resta(dato1,dato2));
    }
    if (opcion==3){
        System.out.println("el resultad de la multiplicacion es : "+cuentas.multiplicacion(dato1,dato2));
    }
    if (opcion==4){
        System.out.println("el resultad de la division es : "+ cuentas.division(dato1,dato2));
    }
    if (opcion >4){
        System.out.println("Debe introducir un numero del 1 al 4 ");
    }


} catch(Exception e) {
    System.out.println("ERROR : " + e);
    e.printStackTrace(System.out);
}
```

Desarrollo de la clase Servidor

En primer lugar, se importa el modulo calculadora, en el estan definidos los metodos que el servidor implementará, luego se importa el servicio de nombres , al igual que el paquete que contiene definidas las excepciones posteriormente, se importa las librerias CORBA.

```
package   calculadora;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.ORB;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
```

En segundo lugar, se definene la clase que implementa las funcionalidades de la calculadora, la misma hereda las funcionalidades de CORBA.

```
class ImplementacionFunciones extends calculoPOA{
private ORB orb;

    public void setORB(ORB orb_val) {
    orb = orb_val;
    }


    public int suma (int a, int b ) {
        return a+b ;
    }

    public int resta(int a, int b) {
        return a-b;
    }

    public int multiplicacion (int a, int b) {
        return a* b;
    }

    public double division (int a, int b) {
        if(b < 1){
            return a/1;
        }
        else {
            return a/b;
        }

    }
}
```

Adicionalmente se crea un metodo setORB, el mismo permite setear el ORB a al servidor.

En tercer lugar, se crea la clase servidor.

```java
public class Servidor {

    public static void main(String[] args) {
        try {

            // Creación e Inicialización del ORB
            ORB orb = ORB.init(args, null);
            POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootpoa.the_POAManager().activate();

            // Se realiza la instancia para implementar las Funciones Remotas
            ImplementacionFunciones calcimp = new ImplementacionFunciones();
            calcimp.setORB(orb);
            org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcimp);
            calculo href = calculoHelper.narrow(ref);
            // obtiene el nombre raiz del contexto
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
            // se establece el nombre que identifica al servidor en el contexto
            String name = "Calculo";
            NameComponent path[] = ncRef.to_name(name);
            ncRef.rebind(path, href);

            System.out.println(" Inicio el servidor :) ");
            // se indica que ORB debe ser iniciada en el ontexto establecido
            orb.run();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Una vez realizado el desarrollo del archivo .idl y de las clases cliente y servidor se procedera a compilar el .idl, el mismo gerenerará las clases necesarias que luego seran instanciadas por el cliente y el servidor.

```
root@Debian:/media/sf_CORBA/Calculadora# idlj -fall calculo.idl
root@Debian:/media/sf_CORBA/Calculadora#
```

Posteriormente, se compila el código de java para verificar que el mismo no posea errores, el comando para ello es el de a continuación.

```
root@Debian:/media/sf_CORBA/Calculadora# javac -d calculadora *.java calculadora
/*.java
Note: calculadora/calculoPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
root@Debian:/media/sf_CORBA/Calculadora#
```

Luego, en la máquina que se empleará como servidor, se ejecutará el siguiente comando:

```
root@Servidor:/media/sf_CORBA/Calculadora/calculadora# orbd -ORBInitialPort 1050 -ORBInitial
Host servermachinename&java calculadora.Servidor -ORBInitialPort 1050
```

```
 Inicio el servidor :)
```

En él se define el puerto en que se deberá ejecutar el servicio de nombres, que en este caso es el 1050, además se indica que aplicación es la que se ejecuta (calculadora.Servidor) y el puerto en el que se ejecutará el servidor (1050).

Luego se debe abrir otra terminal en la misma maquina y se ejecuta el siguiente comando:

```
root@Servidor:/home/sabrina# cd /media/sf_CORBA/Calculadora/calculadora/
root@Servidor:/media/sf_CORBA/Calculadora/calculadora# java calculadora.Cliente
192.168.0.96 -ORBInitialPort 1050
Ingrese un numero:
5
Ingrese un numero:
6
Ingrese una opcion:
Ingrese 1 para la suma:
Ingrese 2 para la resta:
Ingrese 3 para la multiplicacion:
Ingrese 4 para la division:
1
el resultad de la suma es : 11
root@Servidor:/media/sf_CORBA/Calculadora/calculadora#
```

En él, se define la aplicación que se debe conectar con el servidor, a que dirección IP se quiere conectar, (previo se ejecutó un ifconfig para conocer la IP de la maquina) y se especificó en qué puerto se ejecuta el servicio de nombres, en este caso el 1050. Una vez que el cliente pudo establecer la conexión con el servidor, solicita que se ingresen dos números, luego solicita que operación se debe ejecutar, y finalmente realiza las llamadas a los métodos que están implementados en él servidor, imprimiendo en pantalla los resultados.

```
root@Servidor:/media/sf_CORBA/Calculadora/calculadora# java calculadora.Cliente 192.168.0.96 -ORBInitialPort 1050
Ingrese un numero:
60
Ingrese un numero:
9
Ingrese una opcion:
Ingrese 1 para la suma:
Ingrese 2 para la resta:
Ingrese 3 para la multiplicacion:
Ingrese 4 para la division:
3
el resultad de la multiplicacion es : 540
root@Servidor:/media/sf_CORBA/Calculadora/calculadora#
```

```
root@Servidor:/media/sf_CORBA/Calculadora/calculadora# java calculadora.Cliente 192.168.0.96 -ORBInitialPort 1050
Ingrese un numero:
90
Ingrese un numero:
6
Ingrese una opcion:
Ingrese 1 para la suma:
Ingrese 2 para la resta:
Ingrese 3 para la multiplicacion:
Ingrese 4 para la division:
5
Debe introducir un numero del 1 al 4
root@Servidor:/media/sf_CORBA/Calculadora/calculadora#
```

Conclusión

El laboratorio permitió introducir a los estudiantes en el desarrollo de aplicaciones distribuidas basadas en CORBA, ya que es una arquitectura que permite la ejecución de una aplicación que puede ser desarrollada en múltiples lenguajes de programación.

Anexo: Errores

El código de la aplicación está pensado para que se ejecute en red, es por ello que se procedió a clonar la máquina virtual, con el objetivo de tener una maquina cliente. Luego se ejecutó en la maquina servidor la aplicación calculadora.Servidor y una vez que ya estaba disponible, se ejecutó en el cliente calculadora.Cliente que realiza las llamadas al servidor.

Cuando el cliente intenta conectarse con el servidor, este último rechaza la conexión. Ante este error, se probaron diferentes configuraciones, desde establecer direcciones IPs estáticas en cada máquina y configurar las maquina como red interna, hasta modificar el puerto en el que escucha la aplicación, sin embargo, en todos los casos la conexión fallo.

```
root@Servidor:/media/sf_CORBA/Calculadora/calculadora# orbd -ORBInitialPort 1049 -ORBInitial
Host servermachinename&java calculadora.Servidor -ORBInitialPort 1049
[2] 2221
```

```
root@Cliente:/media/sf_CORBA/Calculadora/calculadora# java calculadora.Cliente 192.168.0.98 -ORBInitialPort 1049
jul 23, 2020 8:21:07 PM com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl <init>
ADVERTENCIA: "IOP00410201: (COMM_FAILURE) Connection failure: socketType: IIOP_CLEAR_TEXT; hostname: 192.168.0.84; port: 1049"
org.omg.CORBA.COMM_FAILURE:   vmcid: SUN  minor code: 201  completed: No
        at com.sun.corba.se.impl.logging.ORBUtilSystemException.connectFailure(ORBUtilSystemException.java:2200)
        at com.sun.corba.se.impl.logging.ORBUtilSystemException.connectFailure(ORBUtilSystemException.java:2221)
        at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.<init>(SocketOrChannelConnectionImpl.java:223)
        at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.<init>(SocketOrChannelConnectionImpl.java:236)
        at com.sun.corba.se.impl.transport.SocketOrChannelContactInfoImpl.createConnection(SocketOrChannelContactInfoImpl.java:
        at com.sun.corba.se.impl.protocol.CorbaClientRequestDispatcherImpl.beginRequest(CorbaClientRequestDispatcherImpl.java:1
        at com.sun.corba.se.impl.protocol.CorbaClientDelegateImpl.request(CorbaClientDelegateImpl.java:137)
        at com.sun.corba.se.impl.resolver.BootstrapResolverImpl.invoke(BootstrapResolverImpl.java:99)
        at com.sun.corba.se.impl.resolver.BootstrapResolverImpl.resolve(BootstrapResolverImpl.java:132)
        at com.sun.corba.se.impl.resolver.CompositeResolverImpl.resolve(CompositeResolverImpl.java:47)
        at com.sun.corba.se.impl.resolver.CompositeResolverImpl.resolve(CompositeResolverImpl.java:47)
        at com.sun.corba.se.impl.resolver.CompositeResolverImpl.resolve(CompositeResolverImpl.java:47)
        at com.sun.corba.se.impl.orb.ORBImpl.resolve_initial_references(ORBImpl.java:1177)
        at calculadora.Cliente.main(Cliente.java:18)
Caused by: java.net.ConnectException: Conexión rehusada
        at sun.nio.ch.Net.connect0(Native Method)
        at sun.nio.ch.Net.connect(Net.java:454)
        at sun.nio.ch.Net.connect(Net.java:446)
        at sun.nio.ch.SocketChannelImpl.connect(SocketChannelImpl.java:645)
        at java.nio.channels.SocketChannel.open(SocketChannel.java:189)
        at com.sun.corba.se.impl.transport.DefaultSocketFactoryImpl.createSocket(DefaultSocketFactoryImpl.java:95)
        at com.sun.corba.se.impl.transport.SocketOrChannelConnectionImpl.<init>(SocketOrChannelConnectionImpl.java:207)
        ... 11 more
```

Adicionalmente, se configuro en la ejecución que la aplicación escuche en el puerto por defecto que es el 1049, sin embargo, se obtuvo el mismo resultado.

En la siguiente imagen se puede visualizar que la aplicación está escuchando en el puerto 1049, pero no se logró solucionar el inconveniente.

```
root@Servidor:/home/sabrina# lsof -i:1049
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
orbd    1727 root   16u  IPv6  18085      0t0  TCP *:1049 (LISTEN)
orbd    1727 root   29u  IPv6  19260      0t0  TCP Servidor:1049->localhost:3495
4 (ESTABLISHED)
java    2222 root   21u  IPv6  19259      0t0  TCP localhost:34954->Servidor:104
9 (ESTABLISHED)
root@Servidor:/home/sabrina#
```