PROGRAMACIÓN DE SERVICIOS Y PROCESOS

Aplicaciones de servicios en red

Programación de comunicaciones en red

Servicios en red.

Cuando conectamos dos o más equipos en una red de ordenadores para compartir información y recursos se crean los servicios de red. Estos se configuran en las redes para mantener la seguridad de éstas y mejorar la eficiencia. Además permiten la automatización de determinadas tareas de administración muy complejas facilitando de esta manera la función del administrador de redes.

Entre estos se encuentran:

- *Servicios de archivos*
- *Servicios de impresión*
- *Servicios de mensajería*
- *Servicios de aplicación*
- *Servicios de bases de datos*

Programación de comunicaciones en red

Servicios de archivo.

Incluyen las aplicaciones de red necesarias para almacenar y recuperar datos de archivos. Ayudan a:

- *Hacer más eficiente el uso de hardware de almacenamiento.*
- *Mover con mayor velocidad los archivos de un lugar a otro.*
- *Respaldar datos.*
- *Manejar varias copias del mismo archivo.*

Por ejemplo:

- *Transferencia de archivos*
- *Almacenamiento*

Programación de comunicaciones en red

Servicios de impresión.

Incluyen las aplicaciones de red necesarias para controlar y administrar el acceso a impresoras y fax.

- *Facilitan la reducción del número de impresoras necesarias en una organización.*
- *Mejoran la colocación de impresoras.*
- *Reducen el tiempo de impresión.*
- *Mejoran la gestión de las colas de trabajos de impresión*

Programación de comunicaciones en red

Servicios de mensajería.

Incluyen las aplicaciones necesarias para almacenar, acceder y entregar los mensajes.

- *La organización y mantenimiento de los directorios de información.*
- *El direccionamiento y compartición de datos.*
- *La integración del correo electrónico con los sistemas de voz.*

Programación de comunicaciones en red

Servicios de bases de datos.

Proveen bases de datos a los clientes de la red de manera que éstos puedan almacenar y recuperar los datos para así controlar la manipulación y presentación de éstos.

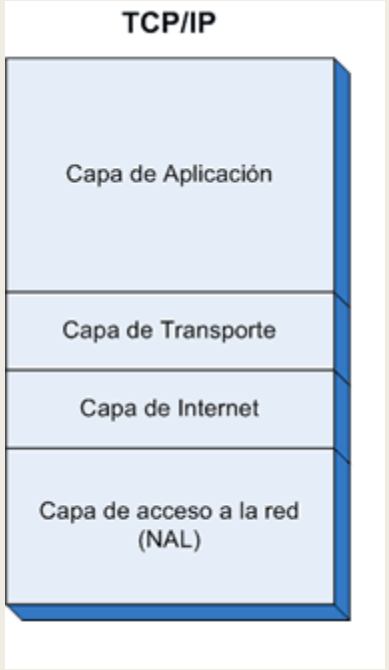
- *Optimizan las computadoras ya que les permite almacenar, localizar y recuperar registros de bases de datos.*
- *Organizar los datos.*
- *Reducir los tiempos de acceso a las bases de datos por parte de los clientes.*

Programación de comunicaciones en red

TCP/IP.

Las capas del modelo TCP/IP son las siguientes:

- Capa de acceso a la red. Se encarga de ofrecer la capacidad de acceder a cualquier red física. Contiene las especificaciones necesarias para la transmisión de datos por una red física.
- Capa de Internet. Se encarga de permitir que los nodos incluyan paquetes en cualquier red y viajen de forma independiente a su destino. Estos paquetes pueden llegar incluso en un orden distinto a como se enviaron. Esta capa define un formato de paquetes y un protocolo oficial denominado IP.

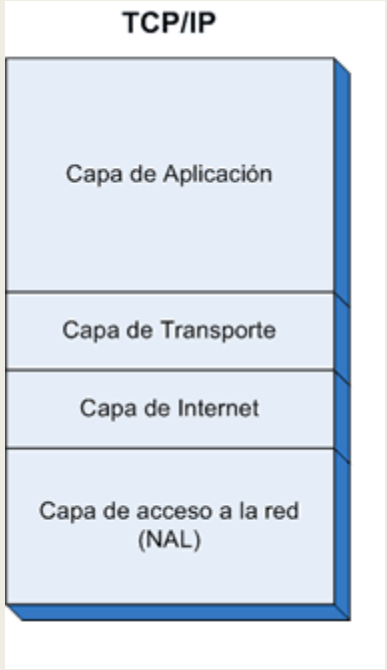


Programación de comunicaciones en red

TCP/IP.

- Capa de transporte. En esta capa se encuentran dos protocolos:
  - *TCP (protocolo de control de la transmisión)*
  - *UDP (protocolo de datagrama de usuario)*

  La principal diferencia entre ambos es que el primero es orientado a la conexión mientras que el segundo es un protocolo sin conexión y no confiable por lo que su uso se limita a aplicaciones que no necesitan control de flujo.
- Capa de aplicación. Contiene los protocolos de más alto nivel como son SMTP, FTP, etc. El software de esta capa se comunica mediante los protocolos de la capa de transporte TCP o UDP.



Programación de comunicaciones en red

Java: Ejemplo TCP/IP. Ejemplo UDP.

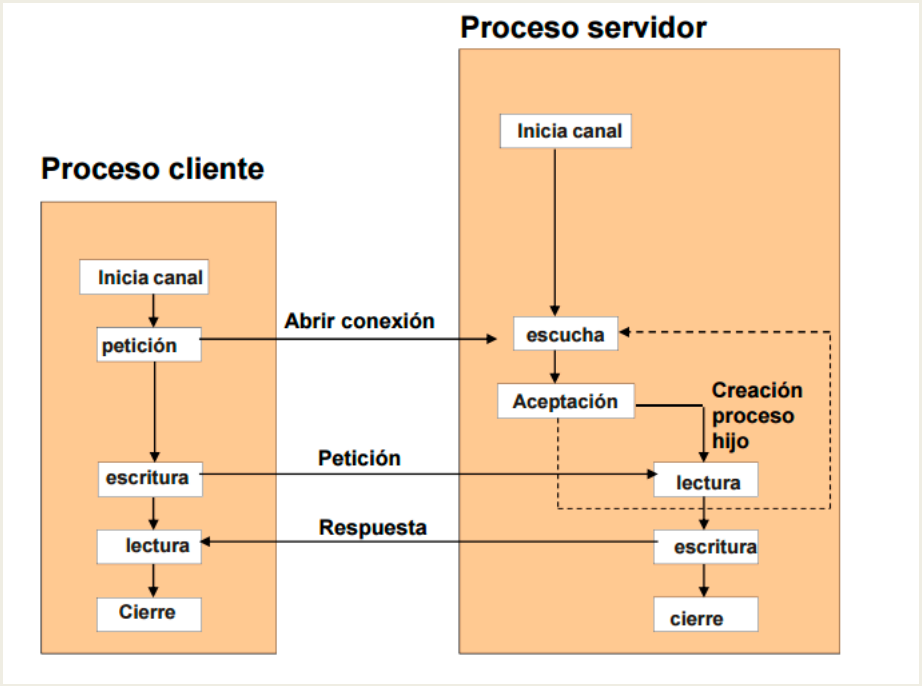
Vemos un ejemplo de ambos protocolos programados en Java

Programación de comunicaciones en red

Programación de aplicaciones cliente y servidor.

El servidor se ejecuta en una máquina específica y tiene un socket asociado a un número de puerto específico, donde espera a que un cliente le envíe una petición. Por la parte de los clientes, como estos saben el nombre de la máquina sobre la que se esté ejecutando el servidor y el n mero de puerto, solicita conexión con el servidor mediante esos parámetros.

Si todo va bien, el servidor acepta la conexión y crea un nuevo socket en un puerto diferente. De esta manera el socket inicial se mantiene a la escucha de nuevas peticiones de clientes y este nuevo socket permite la conexión entre el servidor y el cliente en cuestión.



Programación de comunicaciones en red

Java RMI.

Java RMI es el mecanismo proporcionado por el lenguaje de programación Java para realizar la invocación de métodos de manera remota. Su principal objetivo es permitir a los desarrolladores programar aplicaciones distribuidas en este lenguaje de la misma manera que lo harían para programas no distribuidas.

Java RMI destaca por las siguientes características:

- Facilidad de uso.
- Transparencia.
- Eficiencia.
- Seguridad.
- Permite paso de objetos por referencia.
- Permite la recolección de basura distribuida así como el paso de tipos arbitrarios.

Sin embargo, presenta un gran inconveniente y es que no permite la interacción con aplicaciones desarrolladas en otro lenguaje de programación ya que está integrado de manera estrecha con Java.

Programación de comunicaciones en red

Java RMI.

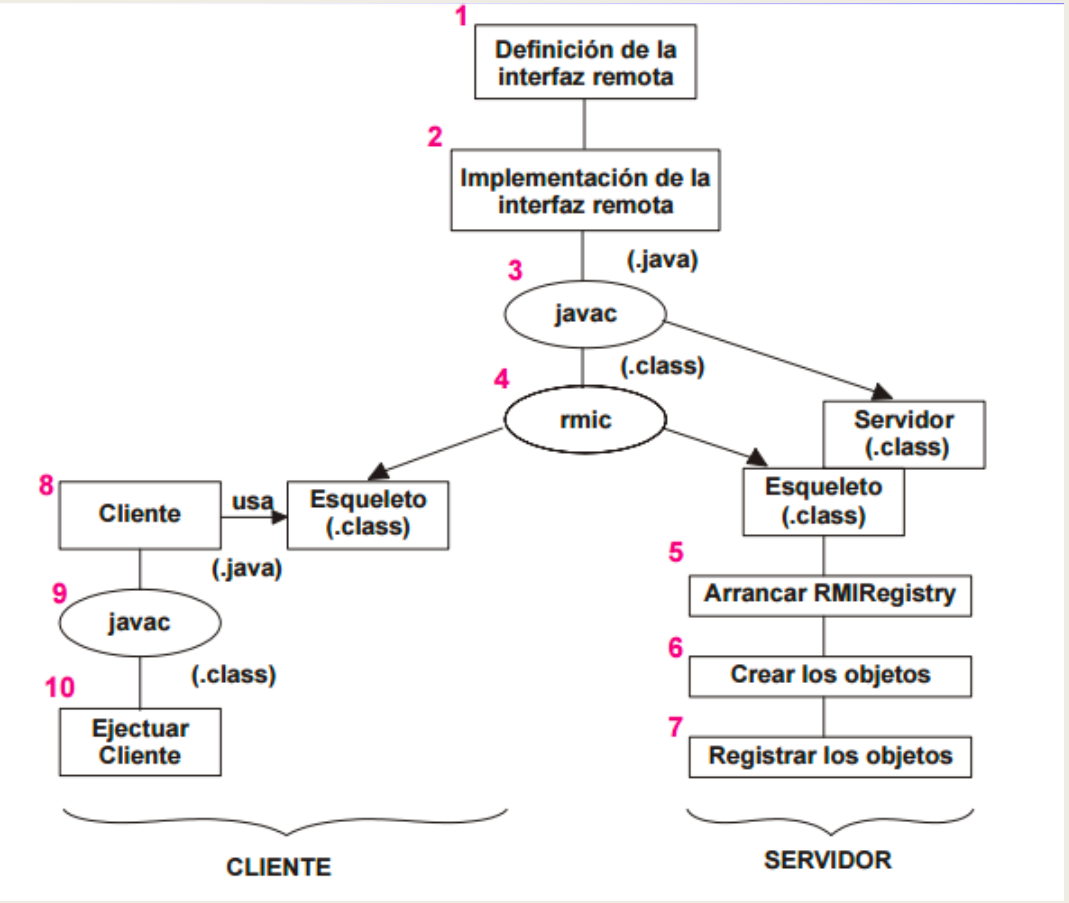
Para la programación de aplicaciones cliente/servidor basadas en invocación de métodos remotos se siguen, de forma general, los siguientes pasos:

1. *Diseño y compilación de la interfaz remota.*
2. *Desarrollo y compilación de la aplicación del servidor.*
3. *Desarrollo y compilación de la aplicación del cliente.*

Programación de comunicaciones en red

Java RMI.

El proceso completo se conforma de:



Programación de comunicaciones en red

Java RMI.

Además hay que destacar que con RMI es necesario tratar mayor número de excepciones que si trabajáramos invocando métodos locales ya que, además de indicar la excepción RemoteException, hay que tener en cuenta:

- *Errores que se puedan producir en la comunicación entre procesos.*
- *Problemas asociados a la propia invocación de los métodos remotos*

Programación de comunicaciones en red

Java RMI.

1. **Diseño y compilación de la interfaz remota.** Para utilizar objetos remotos, tenemos que definir una interfaz con los métodos remotos que queramos utilizar. Esta interfaz debe heredar de la interfaz Remote.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface RMI extends Remote{
    public int sumar(int n1, int n2) throws RemoteException;
}
```

Programación de comunicaciones en red

Java RMI.

2. **Desarrollo y compilación de la aplicación del servidor.** Este deberá instanciar el objeto y ponerlo disponible en la red a través del registro RMI.

```java
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class ServerRMI extends UnicastRemoteObject implements RMI{

    public ServerRMI() throws RemoteException{
        super();
    }

    @Override
    public int sumar(int n1, int n2) throws RemoteException {
        return n1+n2;
    }

    public static void main(String[] args){
        try{
            Registry registro=LocateRegistry.createRegistry(7777);
            registro.rebind("RemotoRMI", new ServerRMI());
            System.out.println("Servidor activo");
        } catch (RemoteException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Programación de comunicaciones en red

Java RMI.

2. **Desarrollo y compilación de la aplicación del cliente.** Este debe buscar el objeto remoto en la red y ejecutar el método correspondiente, en nuestro ejemplo el método sumar().

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Cliente {
    public static void main(String[] args){
        Cliente cliente=new Cliente();
        cliente.connectServer();
    }

    private void connectServer(){
        try{
            Registry registro=LocateRegistry.getRegistry("127.0.0.1",7777);
            RMI interfaz=(RMI)registro.lookup("RemotoRMI");
            int suma=interfaz.sumar(9,6);
            System.out.println("La suma es "+suma);
        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

Programación de comunicaciones en red

Ejercicio

Se pide implementar una aplicación RMI que nos proporcione los siguientes métodos:

- *suma(x,y): que devuelva la suma de los números x e y*
- *resta(x,y): que devuelva la resta de los números x e y*
- *multiplica(x,y): que devuelva la multiplicación de los números x e y*
- *divide(x,y): que devuelva la división de los números x e y*
- *porcentaje(x,y): que devuelva el x% del número y*
- *resto(x,y): que devuelva el resto de la división del número x entre el número y*
- *media(x[]): que devuelva el valor medio de los números que contiene el array x[]*

Programación de comunicaciones en red

Ejercicio

Se pide realizar una aplicación Java RMI cuyo servidor contenga métodos para realizar la conversión entre monedas. Los métodos que se pide implementar son:

- *public float librasEuros (float cantidad) – Devuelve en euros la cantidad que se pasa como parámetro (en libras)*
- *public float eurosLibras (float cantidad) – Devuelve en libras la cantidad que se pasa como parámetro (en euros)*
- *public float dolaresEuros (float cantidad) – Devuelve en euros la cantidad que se pasa como parámetro (en dólares)*
- *public float eurosDolares (float cantidad) – Devuelve en dólares la cantidad que se pasa como parámetro (en euros)*