

Comunicación entre Procesos Distribuidos

M.C. Carlos Rojas Sánchez¹

¹Licenciatura en Informática
Universidad del Mar Campus Puerto Escondido

∴ SISTEMAS DISTRIBUIDOS ∴

Comunicación en los Sistemas Distribuidos

En los sistemas distribuidos, la comunicación siempre se basa en el paso de mensajes de bajo nivel, tal como lo ofrece la red subyacente.

Llamadas a funciones: interfaces de sockets

Sockets API

- Cada tipo de S.O. provee una API de comunicación TCP/IP
- Los sockets presentan aspectos comunes con los Berkeley Sockets
 - Abrir-Leer-Escribir-Cerrar

Sockets

- La relación cliente/servidor es asimétrica. La aplicación requiere saber cuál será su rol (cliente o servidor) antes de iniciar la comunicación.
- La entrada/salida de red puede trabajar orientada a conexión o sin conexión.
- En un entorno de red los nombres son más importantes que en el manejo de archivos o dispositivos.
- Para especificar una conexión de red se requieren más parámetros que para abrir un archivo o dispositivo:
 - {protocolo, dirección_local, proceso_local, dirección_remota, proceso_remoto }
- La interfaz de red debe soportar múltiples protocolos.

Llamadas a funciones remotas: RPC

Llamadas a funciones remotas: RPC

- RPC es una tecnología, tradicionalmente empleada en ambiente UNIX, que permite el desarrollo de sistemas de procesamiento distribuido basados en el paradigma procedimental.
- Una llamada a un procedimiento (función o subrutina) es un método bien conocido para transferir el control de una parte del programa a otra, con un retorno del control a la primera.
- Asociado con la llamada a un procedimiento están el pase de argumentos y el retorno de uno o varios resultados.

RPC

- El “stub” del cliente es empaquetar los argumentos del procedimiento remoto, adecuarlos a algún formato estándar y construir uno o varios mensajes de red.
- El empaquetamiento de los argumentos del procedimiento remoto en mensajes de red se conoce como “marshaling”.

XDR (EXTERNAL DATA REPRESENTATION)

- XDR es un estándar para la descripción y codificación de datos que utiliza un lenguaje cuya sintaxis es similar a la del lenguaje de programación C.
- Es empleado principalmente en la transferencia de información entre diferentes arquitecturas computacionales y se ubica dentro de la capa de presentación del modelo ISO.
- Involucra un mecanismo de tipificado implícito (Implicit Typing), es decir que sólo viaja el valor de la variable por la red.

Tecnologías de sistemas distribuidos

Modelo de objetos distribuidos

Arquitectura RMI (Remote Method Invocation)

- RPC + OO (orientado a objetos)
- Toda aplicación RMI normalmente se descompone en 2 partes:
 - Un servidor, que crea algunos objetos remotos, crea referencias para hacerlos accesibles, y espera a que el cliente los invoque.
 - Un cliente, que obtiene una referencia a objetos remotos en el servidor, y los invoca.

El modelo de objetos

En los lenguajes de programación orientada a objetos, como C++ y Java, un objeto , que encapsula un conjunto de datos y de métodos , se comunica con otros objetos invocando los métodos de éstos, que en general aceptan argumentos y devuelven resultados. Estos lenguajes proporcionan formas para permitir referenciar directamente las variables de los objetos.

El modelo de objetos

Los objetos se acceden por referencia. En la invocación de un método de un objeto (que denominaremos objeto receptor), hay que proporcionar la referencia al receptor, el método y los argumentos de la invocación. Las referencias a objetos pueden asignarse a variables, pasarse como argumentos o devolverse como resultados de una invocación. Una interfaz define el formato de acceso a un conjunto de métodos, es decir, sus nombres, tipos de los argumentos, valores de retorno y excepciones (pero no la implementación de los métodos).

El modelo de objetos distribuidos

En un sistema distribuido, los objetos de las aplicaciones pueden estar repartidos entre los nodos del sistema. Como ocurre en las RPC, la invocación de métodos remotos sigue habitualmente una arquitectura cliente-servidor, donde los objetos receptores se gestionan por los servidores de estos objetos. La invocación mantiene la transparencia en la ubicación del objeto.

El modelo de objetos distribuidos

En el modelo de objetos distribuidos, los procesos constan de objetos que se comunican mediante invocaciones locales o remotas. Las locales son invocaciones a métodos del mismo objeto, mientras que las invocaciones remotas son a métodos de objetos de otros procesos, ya estén en el mismo nodo o en otros nodos del sistema.

Algunos objetos sólo pueden recibir invocaciones locales, mientras otros, los objetos remotos, pueden recibir tanto invocaciones locales como remotas. Para invocar un método de un objeto remoto, un objeto debe tener acceso a la referencia de objeto remoto del receptor, que es un identificador único en el sistema distribuido.

El modelo de objetos distribuidos

Cada objeto remoto posee una interfaz remota que especifica cuáles de sus métodos pueden invocarse remotamente. Estos métodos estarán implementados por la clase del objeto remoto.

- En Java RMI , las interfaces remotas se definen como cualquier otra interfaz de Java, sin más que extender la interfaz Remote .
- CORBA proporciona un lenguaje de definición de interfaces (CORBA IDL) que permite que las clases de los objetos remotos y los programas clientes estén programados en lenguajes diferentes.

Java RMI

Java RMI extiende el modelo de objetos de Java para soportar objetos distribuidos de forma integrada en el lenguaje, haciendo la invocación de métodos remotos transparente salvo por el hecho de que el cliente tenga que tratar las excepciones remotas y el servidor definir como Remote la interfaz del objeto remoto.

Java RMI

Las interfaces remotas se definen extendiendo la interfaz Remote , proporcionada por el paquete java.rmi . Los parámetros de un método son de entrada, y la salida se proporciona en el resultado de la invocación. Los objetos remotos se pasan por referencia y los locales por valor, mediante serialización. Cuando el receptor no dispone de la implementación del objeto que se le pasa por valor, la máquina virtual Java proporciona la descarga automática de la clase correspondiente.

Java RMI

Los nodos que albergan objetos remotos proporcionan un servicio de nombres que almacena las referencias de objetos remotos, el registro de objetos remotos (`rmiregistry`).



En una aplicación distribuida en Java RMI, hay que definir las interfaces remotas e implementar los objetos remotos y los clientes

Java RMI

Una vez compilados los ficheros fuente, la aplicación se monta de la siguiente forma:

- Se crean los proxies (stubs , en terminología Java-RMI) de las clases remotas mediante el compilador de RMI (rmic).
- Se especifica el acceso a las clases para su descarga y se establece una política de seguridad.
- Se pone en marcha el registro de objetos remotos como un proceso del servidor (rmiregistry).
- Se pone en marcha el servicio remoto y se lanzan los clientes.

Bibliografía I

-  Andrew S. Tanenbaum and Maarten Van Steen.
Sistemas Distribuidos: Principios y Paradigmas .
Prentice Hall. 2008, 2ª Edición.
-  Coulouris, G., Dollimore, J., and Kindberg, T..
Distributed Systems: Concepts and Design
Addison-Wesley. 1994, 1ª Edición.