LLAMADA A PROCEDIMIENTO REMOTO RPC

Sanchez Yanque Robin, Cornejo Condori William, Tacuri Sikos Giraldo, Begazo Barrios Richard.
{robinsc1987,, geo20ln}@gmail.com
Universidad Nacional de San Agustin
Arequipa

Resumen—El RPC (del inglés Remote Procedure Call, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente.

Hay distintos tipos de RPC, muchos de ellos estandarizados como pueden ser el RPC de Sun denominado ONC RPC (RFC 1057), el RPC de OSF denominado DCE/RPC y el Modelo de Objetos de Componentes Distribuidos de Microsoft DCOM, aunque ninguno de estos es compatible entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (IDL) que define los métodos exportados por el servidor.

Hoy en día se está utilizando el XML como lenguaje para definir el IDL y el HTTP como protocolo de red, dando lugar a lo que se conoce como servicios web. Ejemplos de éstos pueden ser SOAP o XML-RPC.

Índice de Términos—CORBA, IDL, XML, DCE, OMG, OSF, DCOM, SUN, RMI, JAVA, MIDLEWARE.

1. INTRODUCCION

Un sistema distribuido es un conjunto de computadoras conectadas en red que le dan la sensación al usuario de ser una sola computadora. Este tipo de sistema brinda una serie de bondades tales como: compartición de recursos, la concurrencia, alta escalabilidad, y tolerancia a fallos. A pesar que agregar complejidad al software y disminuir los niveles de seguridad, los sistemas de procesamiento distribuidos brindan una buena relación precio- desempeño y pueden aumentar su tamaño de manera gradual al aumentar la carga de trabajo.

RPC es una tecnología, tradicionalmente empleada en ambiente UNIX, que permite el desarrollo de sistemas de procesamiento distribuido basados en el paradigma procedimental. Con el advenimiento de

implementaciones para plataforma Windows, así como para Java, es posible pensar en RPC como un mecanismo de integración de software heterogéneo.
Los RPC ocultan la comunicación entre procesos, de forma que parezca una llamada a un procedimiento absolutamente normal, es decir, permite a un programa comunicarse con sus partes remotas utilizando el mismo mecanismo (llamada a procedimiento) que utiliza para comunicarse con las locales

2. HISTORIA

La idea de RPC fecha de 1976, cuando fue descrito en el RFC 707. Uno de los primeros usos comerciales de la tecnología fue hecha por la Xerox en el "Courier", de 1981. La primera implementación popular para Unix fue el SunRPC (actualmente llamado ONC RPC), usado como base del Network FileSystem y que aún es usada en diversas plataformas. Otra implementación pionera en Unix fue el Network Computing System (NCS)de la Apollo Computer, que posteriormente fue usada como fundación del DCE/RPC en el Distributed Computing Environment (DCE). Una década después la Microsoft adoptó el DCE/RPC como base para su propia implementación de RPC, MSRPC, a DCOM fue implementada con base en ese sistema. Aún en el mismo periodo de la década de 1990, el ILU de la Xerox PARC y el CORBA ofrecían otro paradigma de RPC basado en objetos distribuidos, con mecanismos de herencia. 1987 Sun Microsystems desarrolla el RPC de cómputo de red abierta (ONC - Open Network Computing) Apollo Computer desarrolla el RPC de su sistema de cómputo en red (NCS - Network Computer System)[1],[2],[6]

- 1989 OSF - Open Software Foundation lanzó una convocatoria para un sistema RPC. Fué elegida la propuesta NCS de compañía HP/DEC. Se conforma el Grupo de Administración de Objetos (OMG - Object Management Group) para coordinar las especificaciones para cómputo distribuido independiente de plataforma y lenguaje.
- 1990 Microsoft basa sus iniciativas de RPC en una versión modificada de DCE/RPC.
- 1991 La OSF libera el DCE 1.0. Se libera CORBA 1.0, con solamente correspondencia del lenguaje C. Se populariza el ORB (Object Request Broker)
- 1996 Microsoft presenta el Modelo de Objeto Componente Distribuido (DCOM - Distributed Component Object Model) CORBA 2.0 es presentado con muchas mejoras en el modelo de cómputo distribuido. El Protocolo de Internet Inter-ORG (IIOP - Internet Inter-ORB) presentado permite la interoperación de múltiples ORBs.
- 1997 Sun Microsystems presenta la herramienta de desarrollo de Java (JDK 1.1) que incluye el Método de Invocación Remota (RMI - Remote Invocation Method). Microsoft anuncia su primera version de COM+, el sucesor de DCOM.
- 1998 Sun Microsystems presenta J2EE (Java 2 Platform Enterprise Edition). Integra el RMI con IIOP, facilitando la interoperación entre sistemas Java y CORBA. Aparece el Protocolo de Acceso de Objeto Simple (SOAP - Simple Object Access Protocol).

De forma análoga, actualmente se utiliza XML como lenguaje de descripción de interfaz y HTTP como protocolo de red para formar servicios Web, cuyas implementaciones incluyen SOAP y XML-RPC

3. OBJETIVOS

- Proporcionar un middleware que simplifique el desarrollo de aplicaciones distribuidas.

- Evitar que el programador tenga que interactuar directamente con el interfaz de Sockets.
- Abstraer (ocultar) los detalles relativos a la red.
- Ofrecer procedimientos que el cliente llama como si fueran procedimientos locales.
- Ofrecer un entorno de programación lo mas similar posible a un entorno no distribuido.
- Implementar la llamada remota mediante un dialogo petición respuesta.

4. LLAMADA A PROCEDIMIENTO REMOTO RPC

4.1. Definición

El RPC (Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC. [2]
Las RPC son muy utilizadas dentro del paradigma cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente. Llamada Remota de Procedimiento es una tecnología de comunicación entre procesos que permite a un programa de ordenador llamar un procedimiento en otro espacio de direccionamiento (generalmente en otro ordenador, conectado por una red). El desarrollador no se preocupa con detalles de implementación de esa interacción remota: del punto de vista del código, la llamada se asemeja a llamadas de procedimientos locales. RPC es una tecnología popular para la implementación del modelo cliente-servidor de computación distribuida. Una llamada de procedimiento remoto es iniciada por el cliente enviando un mensaje para un servidor remoto para ejecutar un procedimiento específico. Una respuesta es retornada al cliente .Una diferencia importante entre llamadas de procedimiento remotas y llamadas de procedimiento locales es que, en el primer caso, la llamada puede fallar por problemas de la red. En ese caso, no hay ni aún garantizaba que el procedimiento fue invocado.[5]

Una definición formal de RPC seria:
"RPC es la transferencia sincrónica de datos y control entre dos partes de un programa distribuido a través de espacios de direcciones disjuntas.”

4.2. Tipos

Hay distintos tipos de RPC, muchos de ellos estandarizados como pueden ser:

- **Sun-RPC (ONC-RPC: Open Network Computing-RPC)** : RPC muy extendido en entornos Unix, infraestructura sobre la que se ejecuta NFS(servicio de sistema de ficheros en red), NIS (servicio de directorio).
- **DCE/RPC (Distributed Computing Environmen RPC)**: RPC definido porla Open Software Foundation
- **Java-RMI:** Invocación de métodos remotos en Java
- **CORBA (Common Object Requesting Broker Architecture)** : soporta la invocación de métodos remotos bajo un paradigma orientado a objetos en diversos lenguajes
- **SOAP (Simple Object Access Protocol):** protocolo RPC basado en el intercambio de datos

(parámetros+resultados) en formato XML

- **DCOM (Distributed Component Object Model)** : Modelo de Objetos de Componentes Distribuidos de Microsoft, con elementos de DCE/RPC
- **.NET Remoting** : Infraestructura de invocación remota de .NET.

5. CARACTERÍSTICAS DEL RPC

- Orientadas a modelo cliente- servidor
- RPC se encarga automáticamente de:
  - Marshalling (Aplanamiento) y unmarshalling
  - Control de servicio concurrente
- Normalmente implican comunicación uno a uno
  - En algunos sistemas Multicast y Broadcast RPC (p. ej. RPC de Sun)
- Comportamiento ante fallos garantizado por RPC
  - P.ej. RPC de DCE asegura por defecto como mucho una vez
- Además permite marcar una función de servicio como idempotente lo que posibilita repetirla hasta que se complete (al menos una vez)
- Estudio más detallado de dos aspectos:
  - Direccionamiento
  - Lenguaje de definición de interfaces (IDL)

6. FUNCIONAMIENTO

Para comprender el funcionamiento de RPC, es importante entender el funcionamiento de una llamada convencional a un procedimiento(es decir, en una sola maquina).

Consideramos una llamada como
Count=read (fd, buf, nbytes);

En donde fd es un entero, buf es un arreglo de caracteres y nbytes es otro entero. Si la llamada se hace en el programa principal la pila se verá como en la figura (1a) antes de la llamada. Para ser esta, quien la hace coloca los parámetros en la pila, en orden, el último en primer lugar, ver figura (1b). Después de que read termine su ejecución, se coloca el valor de retorno en un registro, elimina la dirección y transfiere de nuevo el control a quien hizo la llamada. Este último elimina los parámetros de la pila y regresa a su estado original, como se ve en la figura (1c).[3]

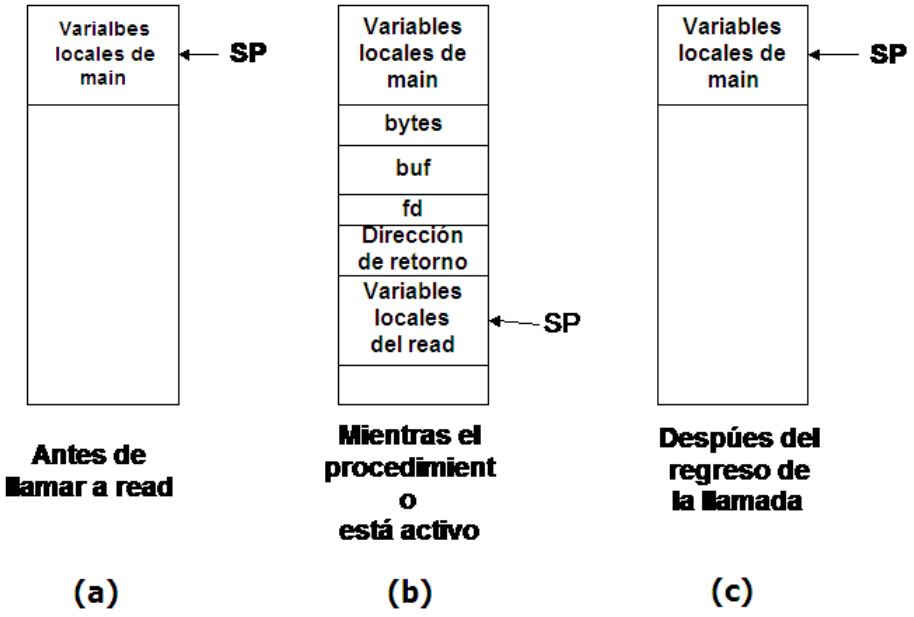


Figura 1

La idea detrás de RPC es que una llamada a un procedimiento se parezca lo más posible a una llamada local. Lo que se busca es que la RPC sea transparente, es decir, el procedimiento que hace la llamada no debe ser consciente de que el procedimiento llamado se ejecuta en una maquina distinta, o viceversa.

En un sistema tradicional (con un procesador) el ligador extrae la rutina read de la biblioteca y lo inserta en el programa objeto. Este procedimiento coloca los parámetros en registros y después hace una llamada al sistema read mediante un señalamiento al núcleo. Aunque read realiza un señalamiento al núcleo, se le llama de manera usual, al colocar los parámetros en la pila. Así el programador no tiene que saber sobre dicho señalamiento.

RPC logra su transparencia de manera análoga. Si read es un procedimiento remoto, se coloca en la biblioteca una versión distinta a send, llamada resguardo o stub del cliente. Como el original, también se le llama mediante la secuencia de llamada de la figura 1 y también este hace un señalamiento al núcleo. Pero a diferencia del original no coloca los parámetros en registros y le pide al núcleo que le proporcione datos, si no que empaca los parámetros en un mensaje (marshalling) y le pide al núcleo que envié el mensaje al servidor (Ver figura 2). El envío de mensajes se realiza mediante la llamada que hace el resguardo del cliente a un procedimiento denominado send, posteriormente este llama a receive (procedimiento para la recepción) y se bloquea hasta que regrese la respuesta.[2]

Cuando el mensaje llega al servidor, el núcleo lo transfiere a un resguardo del servidor el cual estaba bloqueado en espera de que le llegara un mensaje, el resguardo del servidor desempaca los parámetros del mensaje y después llama al procedimiento del servidor de manera usual (como en se muestra en la figura 1). Posteriormente el resguardo del servidor empaca el resultado en un mensaje y llama a send para regresarlo al cliente y a receive para esperar el siguiente mensaje.

Cuando el mensaje regresa a la maquina cliente, el núcleo lo dirige hacia el resguardo quien después de eliminar su bloqueo, examina el mensaje, desempaca el resultado, lo copia a quien hizo la llamada y regresa de manera usual. Todo el proceso anteriormente descrito se puede observar en la siguiente figura

(2).

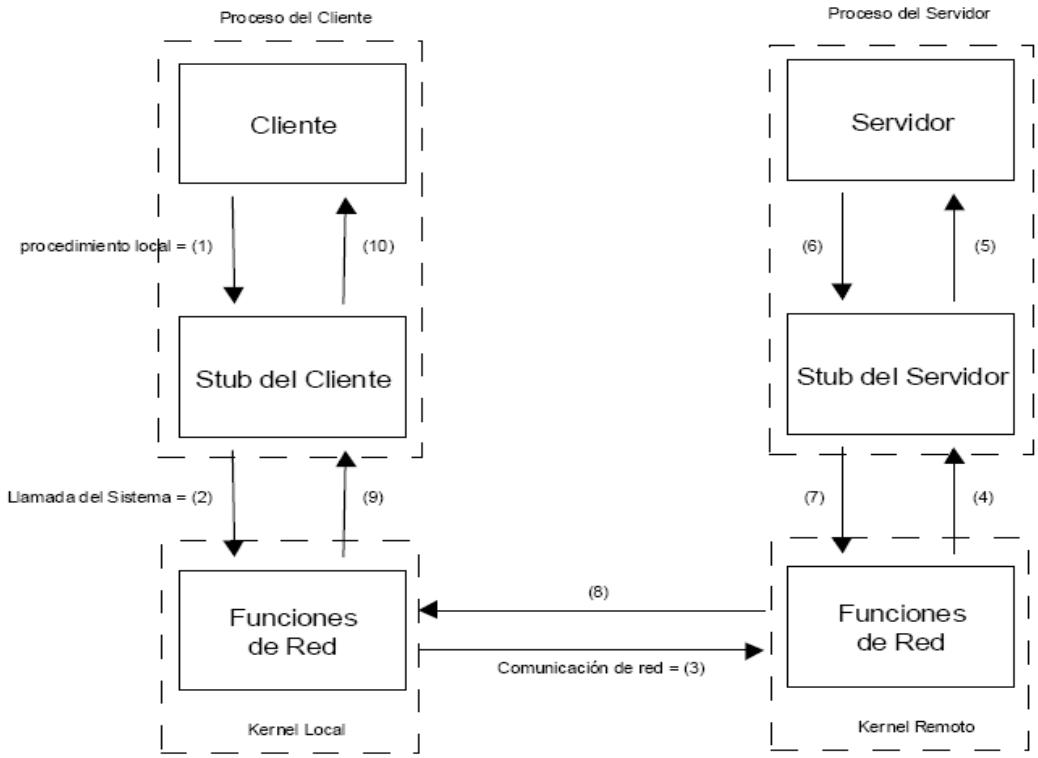


Figura 2

En resumen, una llamada a un procedimiento remoto se realiza mediante los siguientes pasos:[2],[7]

1. El cliente llama a un procedimiento local llamado “stub” del cliente, el cual aparenta ser el procedimiento servidor que el cliente desea llamar. El propósito del “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”.
2. Estos mensajes son enviados por el “stub” del cliente al sistema remoto, lo cual requiere una llamada del sistema.
3. Los mensajes son transferidos al sistema remoto empleando protocolos con o sin conexión.
4. Un procedimiento “stub” del servidor espera en el sistema remoto la solicitud del cliente. Desempaqueta los argumentos de los mensajes de red y si es necesario realiza alguna conversión.
5. El “stub” del servidor realiza la llamada al procedimiento local que realmente invoca la función del servidor y le pasa los argumentos transferidos a través de la red desde el “stub” del cliente.
6. Cuando el procedimiento del servidor termina, éste le regresa el control al “stub” del servidor devolviendo los resultados obtenidos.
7. El “stub” del servidor adecua el formato de tales resultados, si es necesario, y los empaqueta en mensajes de red para ser devueltos al “stub” del cliente.
8. Los mensajes son transmitidos al “stub” del cliente.
9. El “stub” del cliente lee los mensajes recibidos.
10. Luego de posiblemente convertir los valores de retorno, el “stub” del cliente retorna finalmente dichos resultados a la función del cliente haciendo parecer un retorno normal de función.

6.1. FUNCIONAMIENTO

- El stub del cliente: se encarga de empaquetar los parámetros y la solicitud, enviarlos al intermediario en el servidor, y luego esperar la respuesta, desempaquetarla y entregarla a la aplicación.

- El programa principal del servidor (que incluye el stub y el dispatcher). se encarga de recibir peticiones, desempaquetar los parámetros, invocar la función solicitada, pasarle los parámetros, luego obtener el resultado, empaquetarlo y enviarlo al cliente.[2]

- Las rutinas de serialización de datos: Se debe tomar en cuenta que las máquinas cliente y servidor puedan ser de arquitectura diferente (y no compatible).

- Servicio de binding: Responsable de la transparencia de localización, gestiona la asociación entre el nombre del procedimiento remoto (y su versión) con su localización en la maquina servidor (dirección, puertos, skeleton, etc). Realiza la búsqueda del skeleton de la implementación concreta del procedimiento remoto llamado por un cliente. Ejemplos: portmapper en Sun-RPC, protocolo UDDI en servicios web, rmiregistry en Java-RMI.[6]

6.2. CARACTERÍSTICAS DEL STUB

La base del mecanismo RPC consiste en la introducción de "representantes" que "hacen como si" fuesen el cliente/servidor[1],[5].[6]

- En el lado cliente, el representante del servidor se denomina *Stub* (o *Proxy*)
- El *stub* es quien proporciona transparencia en la invocación del cliente
- El *stub* debe poseer llamadas con la misma declaración (forma) que el servidor
- El cliente invoca las llamadas del *stub* "como si" fuese el servidor
- El *stub*, a través de un protocolo RPC y con unos mecanismos de aplanamiento, envía un mensaje al extremo remoto solicitando la "ejecución real" de la llamada
- El *stub*, a través de un protocolo RPC y con unos mecanismos de desaplanamiento, recibe un mensaje del extremo remoto y recupera el "resultado" de la invocación.
- El *stub* oculta los detalles de referencia del objeto remoto. Es decir, debe saber en qué dirección IP y en qué puerto hay que contactar con el extremo remoto
- Cada procedimiento que el cliente quiera invocar a través de RPCs necesita su propio *stub*

6.3. CARACTERÍSTICAS DEL SKELETON

- En el lado servidor, el representante del cliente se llama Skeleton
- El skeleton es quien proporciona transparencia en el lado del servidor
- El skeleton debe conocer las llamadas ofrecidas por el servidor
- El skeleton, a través de un protocolo RPC y con unos mecanismos de desaplanamiento, recibe un mensaje del extremo remoto solicitando la "ejecución real" de la llamada
- El skeleton invoca la llamada del servidor "como si" fuese el cliente

- El skeleton, a través de un protocolo RPC y con unos mecanismos de aplanamiento, envía un mensaje al extremo remoto indicando el "resultado" de la invocación.
- Cada procedimiento que el servidor exporte a través de RPCs requiere su propio skeleton

7. VENTAJAS Y DESVENTAJAS

7.1. VENTAJAS

- El mecanismo de enviar mensajes de solicitud y esperar la respuesta queda oculto debajo de un esquema fácil de entender: llamar a un procedimiento, el cual devuelve resultados.
- RPC ofrece un mecanismo de representación de datos independiente de la máquina conocido como XDR (External Data Representation).
- RPC facilita la definición del protocolo de comunicación (al nivel de aplicación).

7.2. VENTAJAS

- No se pueden transportar punteros.
- Se pueden presentar una serie de fallos en los componentes de comunicación (Cliente, servidor, red)
    - El cliente no puede localizar al servidor
    - Se pierde el mensaje de solicitud del cliente al servidor
    - Se pierde el mensaje de respuesta del servidor al cliente
    - El servidor falla antes de escribir la solicitud
    - El cliente falla después de enviar su solicitud

8. PORT MAPPER

- Es un servicio que mapea un número de programa y versión de RPC a un puerto de UDP o TCP. Portmapper tiene asignado el puerto \well Known" 111.[3]
- Cada vez que se arranca uno de los servicio RPC en un servidor, se registra en el servicio Portmapper de dicho host asociándole un determinado valor de puerto a dicho servicio.
- Un cliente RPC contacta con el servicio Portmapper para obtener el valor del puerto para un determinado RPC.
- Después envía el RPC a dicho puerto.

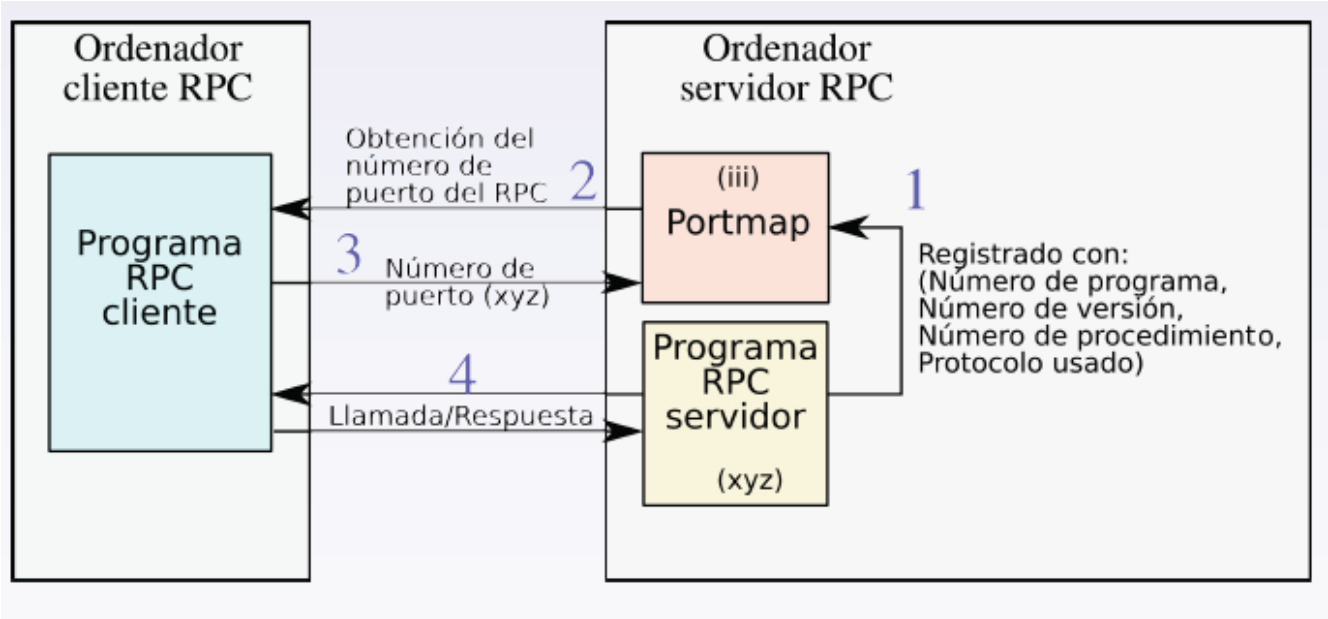


Figura 3

9. BIG ENDIAN Y LITTLE ENDIAN

Existen varios aspectos que hacen difícil la comunicación entre aplicaciones distribuidas. Cada arquitectura puede definir formas diferentes de almacenar sus datos en la memoria (por ejemplo, las little-endian versus las big-endian). Un entero de 32 bits enviado entre máquinas de tipo diferente será interpretado erróneamente sino se convierte. [3]
El problema de comunicar máquinas de arquitectura incompatible (por ejemplo, una PC con una Sparc) se resuelve convirtiendo los parámetros a un formato independiente de la máquina en el envío, y traduciendo del formato independiente al nativo (propio de la máquina) en la recepción.

Vamos a verlo usando como ejemplo el número 5. Si lo expresamos en binario con 4 bits, su equivalencia es 0101. Si lo hacemos con 8 bits, sería 00000101. Pero si usamos más de un byte para representarlo, por ejemplo porque se tratara de un número "entero corto", usando la nomenclatura del lenguaje C, tenemos que el número 5 equivaldría a 16 dígitos binarios: 00000000 00000101.

Por tanto, ahora este número 5 queda representado como 2 bytes. El de la izquierda es el byte más significativo (MSB) y el de la derecha es el bit menos significativo (LSB). Estos nombres vienen de que un cambio en el byte de la derecha hace que el número cambie poco (por ejemplo, 00000000 00000100 = 4 en decimal), mientras que un cambio en el byte de la izquierda provoca cambios mucho más grandes (00000001 00000101 = 261 en decimal).[1],[3]

Pues bien, según la arquitectura del ordenador puede ocurrir que esos dos bytes se guarden en el orden que hemos visto: 00000000 00000101 (primero el MSB y luego el LSB) o al contrario (00000101 00000000). Lo primero, que a priori es lo que podría parecer más natural, es lo que se hace en

procesadores como los de la familia PowerPc (usados en muchos ordenadores Apple, por ejemplo). Lo segundo, aunque parezca más enrevesado, es lo que se hace en la mayoría de procesadores de Intel, como los usados en los ordenadores PC.

El primer formato (MSB LSB) es lo que se conoce como Big Endian, porque eso de que el extremo más grande aparece en primer lugar. El segundo formato (LSB MSB) es lo que se conoce como Little Endian, porque se almacena primero el dato más pequeño.

Afectaría a un programador en que muchas veces debería saber con qué plataforma se ha creado un fichero de datos, para poderlo interpretar correctamente.

Por ejemplo, si leemos un fichero que contiene un dato "entero corto", de 2 bytes, formado por la secuencia 00000101 00000000, y ese fichero se ha creado con un ordenador basado en un Pentium u otro procesador de Intel (Little Endian), podemos suponer que ese número es un 5. Por el contrario, si el dato se ha creado desde un equipo Apple clásico, la secuencia 00000101 00000000 estaría en formato Big Endian, luego equivaldría al número 1280.

En general, en caso de que un fichero contenga datos numéricos de más de un byte, deberíamos saber si están almacenados en formato Little Endian o Big Endian, e intercambiar el orden de los bytes después de leer, si fuera necesario.

Como primera curiosidad, existen arquitecturas que permiten escoger la "endianness" que se prefiere usar (como IA64, MIPS y ARM) y que reciben el nombre de "bi-endian". En estos sistemas, normalmente este cambio se puede hacer por software (al arrancar el equipo,por ejemplo), pero en algún caso se ha de realizar por hardware (como podría ser cambiando un jumper en la placa base).

Para el caso del problema con complemento a 1 y 2 es en general el mismo problema que se presenta con "Big Endian" y "Little Endian"

10. TRANSPARENCIA DE ACCESO Y LOCALIZACION

10.1. ACCESO

Un servicio proporcionado por un servidor no es más que un conjunto de operaciones disponibles para los clientes. El acceso al servicio se realiza mediante un protocolo de peticiones respuesta con llamadas bloqueantes. Ejemplo: Un servicio de ficheros. El servidor mantiene como recurso compartido los ficheros. Sobre el recurso compartido se pueden realizar diversas operaciones: Crear, Abrir, Leer, etc.[4]

Los mecanismos RPC persiguen que los clientes se abstraigan e invoquen procedimientos remotos (operaciones) para obtener servicios. Así, el procedimiento llamado se ejecuta en otro proceso de otra maquina (servidor). El objetivo de RPC es mantener la semántica de la llamada a procedimiento normal en un entorno de implementación totalmente distinto. La ventaja esta en que el desarrollador se preocupa de los interfaces que soporta el servidor. Para especificar dichos interfaces se dispones de un IDL (lenguaje de definición de interfaces).

Los sistemas RPC disponen de mecanismos de RPC integrados en un lenguaje de programación

particular que incluye además una notación para definir interfaces entre clientes y servidores (IDL específico). Un IDL permite definir el nombre de las operaciones soportadas por el servidor y sus parámetros (tipo y dirección). También se deben proveer mecanismos para manejo de excepciones, garantizar la ejecución de las operaciones, así como la detección de fallos. Todo ello de la forma más transparente posible.[5]

10.2. LOCALIZACION

Sun RPC requiere encapsular las funciones del servidor en un programa. Un programa puede tener una o varias versiones, las cuales van a contener las funciones del servidor. Cada componente posee un identificador único el cual permitirá al cliente realizar un llamado a una función específica.

```
Ej.:
program  NOMBRE_PROGRAMA {
    version VERSION_PROGRAMA {
        int incrementa (int) = 1;
    } = 1;
} = 0x20000001;
```

El ejemplo comienza con la palabra "program" y el nombre del programa. Al final todo se iguala a un número (= 0x20000001;). Este número debe ser único para cada programa. De esta forma podemos tener varios servidores corriendo a la vez en el mismo ordenador y los clientes tendrán una forma de referenciarlos. [6],[8]

Dentro de las llaves del programa va "version" con el nombre de la version y, después de la estructura de llaves, igualado a un número de versión (= 1;). Esto permite tener corriendo varias versiones del mismo programa (y de las mismas funciones) a la vez. Los clientes pueden indicar a qué versión quieren llamar. Finalmente, dentro de las llaves de versión, van los prototipos de las funciones. Cada función se iguala a un número (= 1;) distinto, de forma que el cliente referenciará a la función por su número.

- El cliente debe poder contactar un programa remoto. Para ello necesita saber el puerto por el cual escucha el servidor (intermediario) de ese programa. Pero el cliente no puede conocerlo porque ese puerto es asignado cuando el servicio arranca.[7]

- Para resolver este problema, Sun RPC utiliza un servidor por máquina que contiene las asignaciones de programas con sus versiones a puertos en esa máquina, el portmapper. Este está asignado a un puerto conocido (111) y permite a los servicios activos en esa máquina registrarse para ser accesibles, así los clientes pueden contactar el portmapper y preguntarle a que puerto está asignado un determinado programa, para que posteriormente se pueda establecer la conexión con el servidor y se realice la solicitud de servicios (Ver figura 4).

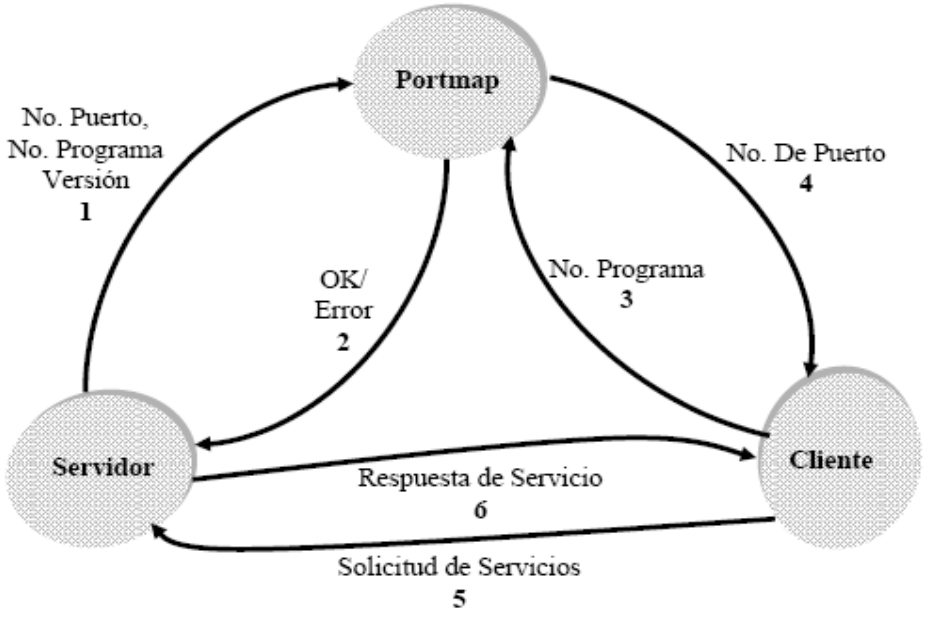


Figura 4

El proceso puede entenderse aun más con la siguiente figura (5):

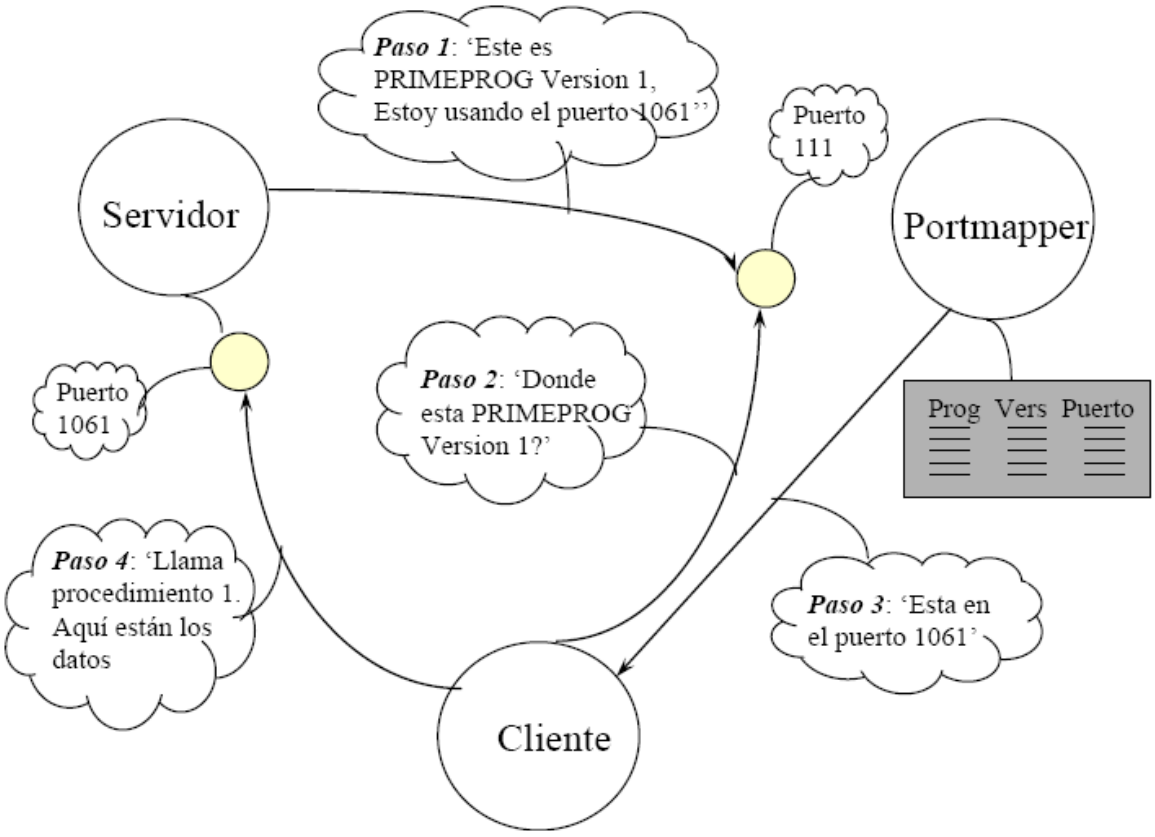


Figura 5

11. RPC SINCRONO Y RPC ASINCRONO

11.1. RPC SINCRONO

Cuando un cliente manda una petición al servidor este queda bloqueado esperando la respuesta que la va enviar el receptor antes de realizar otro proceso. [7]

La RPC síncrona no es capaz de explotar por completo el paralelismo inherente a las aplicaciones distribuidas; esto limita el tipo de interacción que las aplicaciones distribuidas pueden realizar y así obtienen un rendimiento menor.

11.2. RPC ASINCRONO

Cuando el cliente le envía una petición al servidor este no se queda bloqueado en espera de la respuesta del servidor y puede seguir su ejecución inmediatamente después de enviar la petición

Para ofrecer una mayor flexibilidad se implementaron varios servicios de RPC asíncrona que consiguen un grado mayor de paralelismo. [7]

12. COMPARACIONES

12.1. RPC VS RMI

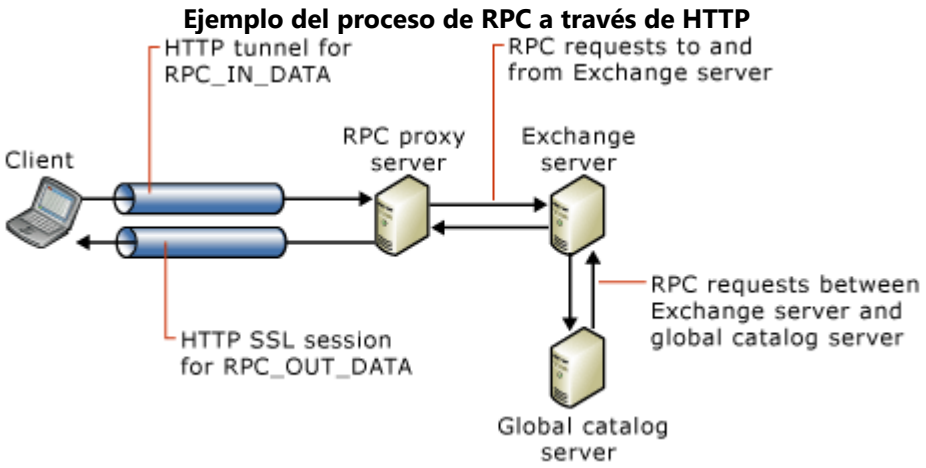
RPC	RMI
No es orientado a objeto	Orientado a objeto
Menor Seguridad	Mayor seguridad
Fácil la programación	Programación complicada
Mejor rendimiento	Menor rendimiento

12.2. RPC VS CORBA

RPC	CORBA
Una plataforma	multiplataforma
Un solo lenguaje	Varios lenguajes
No es orientado a objeto	Orientado a objeto
Fácil la programación	Programación complicada

13. EJEMPLO DEL PROCESO RPC A TRAVÉS DE HTTP

En la figura siguiente se muestran las interacciones entre el cliente Microsoft® Office Outlook®, el servidor proxy RPC y los servidores de servicios de fondo



En la figura siguiente se muestran las interacciones entre el cliente de Microsoft® Office Outlook®, el servidor proxy RPC y los servidores de servicios de fondo. En este ejemplo se da por supuesto que el servidor de carpetas públicas del usuario es el mismo servidor que el servidor de buzón del usuario. En este ejemplo también se da por supuesto que se está ejecutando el Service Pack 1 (SP1) de Microsoft Exchange Server 2003 en los servidores de Exchange.

Paso 1

El equipo cliente ha determinado que se conectará al servidor de Exchange utilizando RPC a través de HTTP. El equipo cliente inicia dos sesiones HTTP en el servidor de Exchange para cada solicitud de RPC que deba enviar. El equipo cliente inicia una sesión HTTP para solicitudes RPC en el servidor de Exchange y otra sesión HTTP para respuestas del servidor de Exchange.

El equipo cliente envía las solicitudes de RPC iniciales al servicio de referencia de DSProxy del servidor de Exchange que solicita una referencia del directorio. El puerto de servicio de referencia es 6002.

Paso 2

El servidor proxy RPC extrae la solicitud RPC de la sesión HTTP y la reenvía al puerto 6002 del servidor de Exchange. El servidor de Exchange responde a la solicitud de referencia del directorio con él mismo como destino. Este método es exclusivo de RPC a través de HTTP. Si el equipo cliente no utiliza RPC a través de HTTP, el servidor de Exchange respondería con un servidor de catálogo global. Sin embargo, cuando el equipo cliente utiliza RPC a través de HTTP, el servidor de Exchange reconoce la solicitud del equipo cliente como RPC a través de HTTP. El equipo cliente no puede acceder directamente a un servidor de catálogo global cuando el equipo cliente

utiliza RPC a través de HTTP. Por lo tanto, el servidor de Exchange responde al equipo cliente con él mismo, el servidor de Exchange, como servidor para utilizarlo para búsquedas del directorio.

Paso 3

El equipo cliente inicia dos sesiones HTTP para el servicio proxy de DSProxy en el servidor de Exchange. El servicio proxy está en el puerto 6004. El equipo cliente inicia una sesión HTTP para solicitudes de RPC en el servidor y otra sesión HTTP para solicitudes de RPC desde el servidor.

Paso 4

El servidor proxy RPC extrae la solicitud RPC de la sesión HTTP y la reenvía al puerto 6004 del servidor de Exchange.

Paso 5

El servidor local de Exchange reenvía la solicitud del directorio a un servidor de catálogo global. El servidor de catálogo global responde al servidor de Exchange que tiene la información de directorio apropiada.

Paso 6

El servidor de Exchange envía la información del directorio que ha recibido del servidor de catálogo global al equipo cliente.

Paso 7

El equipo cliente inicia dos sesiones HTTP para el servicio Almacén de información de Microsoft Exchange en el servidor de Exchange. El servicio Almacén de información de Microsoft Exchange está en el puerto 6001. El equipo cliente inicia una sesión HTTP para solicitudes RPC en el servidor y otra sesión HTTP para solicitudes RPC desde el servidor.

Paso 8

El servidor proxy RPC extrae la solicitud RPC de la sesión HTTP y la reenvía al puerto 6001 del servidor de Exchange.

Los pasos 7 y 8 se repiten para conexiones de almacenes adicionales, como el acceso a datos de carpetas públicas.

14. CONCLUSIONES

- RPC es una tecnología, tradicionalmente empleada en ambiente UNIX, que permite el desarrollo de aplicaciones cliente servidor basado en el paradigma procedimental.

- Se concibe RPC como una tecnología de integración entre plataformas diferentes de hardware y software.
- RCP realiza el requerimiento del cliente en una máquina que esta disponible para ejecutar el proceso, siendo transparente para el cliente.

15. REFERENCIAS

[1] Llamada de Procedimiento Remoto, http://lucas.hispalinux.es/Manuales-LuCAS/GARL2/garl2/x-087-2-appl.rpc.html
[2] Llamada de Procedimiento Remoto, *www.tamps.cinvestav.mx/~vjsosa/clases/sd/RPC_Todo.pdf*
[3] Llamada a Procedimiento Remoto, http://robertomc.8k.com/rpc1.htm
[4] A. Tanenbaum, et al., "Sistemas Operativos. Diseño e implementación", 2da. Edición, Prentice Hall, México, 1998, ISBN: 970-17-0165-8.
[5] Tanenbaum, Andrew (1996). Sistemas Operativos Distribuidos. México, Prentice Hall.
[6] Remote Procedure Call, http://grasia.fdi.ucm.es/sensei/docs/sensei.pdf
[7] Remote Procedure Call, http://es.wikipedia.org/wiki/Java_Remote_Method_Invocation