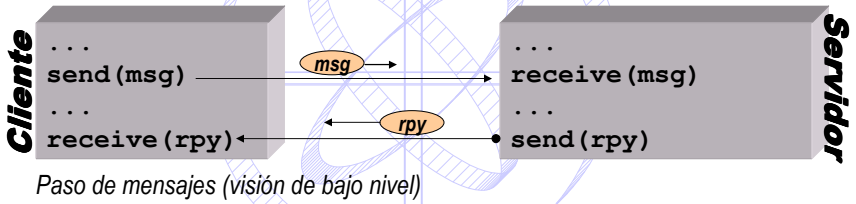
Sistemas Distribuidos

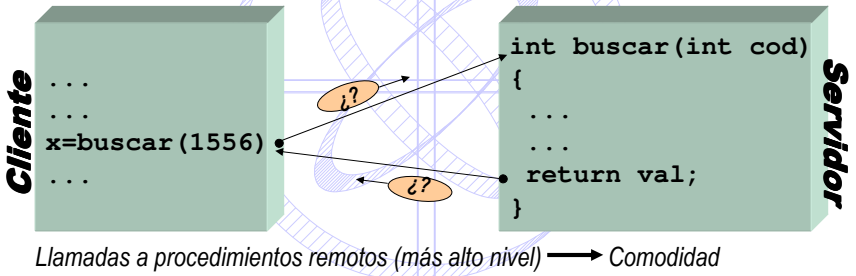
Llamadas a Procedimientos Remotos
Sun RPCs

Llamadas a Procedimientos Remotos



Paso de mensajes (visión de bajo nivel)

Llamadas a Procedimientos Remotos



Llamadas a procedimientos remotos (más alto nivel) → *Comodidad*

Llamadas a Procedimientos Remotos

- Remote Procedure Call: RPC.
- Evolución:
  - Propuesto por Birrel y Nelson en 1985.
  - Sun RPC es la base para varios servicios actuales (NFS o NIS).
  - Llegaron a su culminación en 1990 con DCE (Distributed Computing Environment) de OSF.
  - Han evolucionado hacia orientación a objetos: invocación de métodos remotos (CORBA, RMI).

Funcionamiento General de RPC

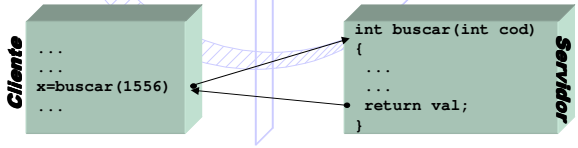
- Cliente:
  - El proceso que realiza una la llamada a una función.
  - Dicha llamada empaqueta los argumentos en un mensaje y se los envía a otro proceso.
  - Queda la espera del resultado.
- Servidor:
  - Se recibe un mensaje consistente en varios argumentos.
  - Los argumentos son usados para llamar una función en el servidor.
  - El resultado de la función se empaqueta en un mensaje que se retransmite al cliente.

Funcionamiento General de RPC

- Objetivo: acercar la semántica de las llamadas a procedimiento convencional a un entorno distribuido (transparencia).

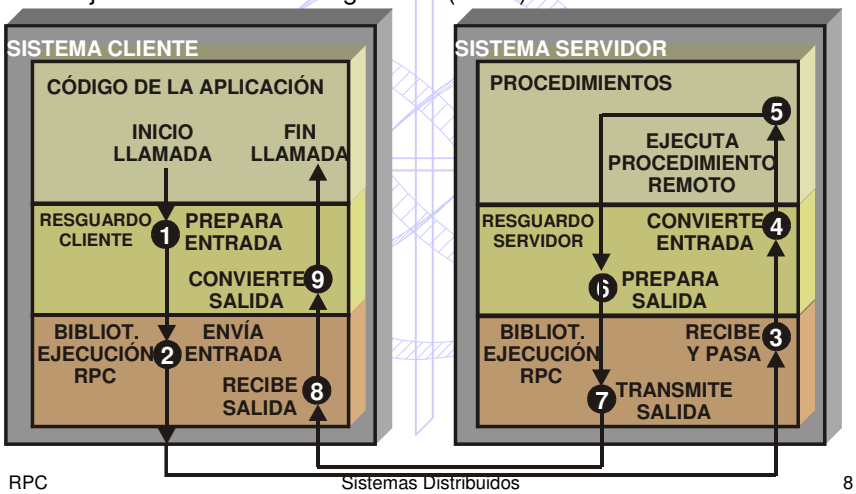
Elementos Necesarios

- Código cliente.
- Código del servidor.
- Formato de representación.
- Definición del interfaz.
- Localización del servidor.
- Semánticas de fallo.



Código Cliente/Código Servidor

Las funciones de abstracción de una llamada RPC a intercambio de mensajes se denominan resguardos (*stubs*).



Resguardos (*stubs*)

- Se generan automáticamente por el software de RPC en base a la interfaz del servicio.
  - Son independientes de la implementación que se haga del cliente y del servidor.
  - Sólo dependen de la interfaz.

Resguardos (*stubs*)

- Tareas que realizan:
  - Localizan al servidor.
  - Empaquetan los parámetros y construyen los mensajes.
  - Envían el mensaje al servidor.
  - Espera la recepción del mensaje y devuelven los resultados.
- Se basan en una librería de funciones RPC para las tareas más habituales.

Formato de Representación

- Una de las funciones de los resguardos es empaquetar los parámetros en un mensaje: aplanamiento (*marshalling*).

Formato de Representación

- Problemas en la representación de los datos
  - Servidor y cliente pueden ejecutar en máquinas con arquitecturas distintas.
    - Big Endian o Little Endian
  - XDR (*external data representation*) es un estándar que define la representación de tipos de datos.
  - Pasos de parámetros (entrada/salida):
    - Problemas con los punteros: Una dirección sólo tiene sentido en un espacio de direcciones.

Definición de Interfaces: IDL

- IDL (Interface Definition Language) es un lenguaje de representación de interfaces:
  - Hay muchas variantes de IDL:
    - Integrado con un lenguaje de programación (Cedar, Argus).
    - Específico para describir las interfaces (RPC de Sun y RPC de DCE).
  - Define procedimientos y argumentos (No la implementación).
  - Se usa habitualmente para generar de forma automática los resguardos (stubs).

Definición de Interfaces: IDL

```
Server ServidorTickets
{
      procedure void reset();
      procedure int  getTicket(in string ident);
      procedure bool isValid(in int ticket);
}
```

Localización del Servidor

- La comunicación de bajo nivel entre cliente y servidor por medio de paso de mensajes (por ejemplo sockets)
- Requiere
  - Localizar la dirección del servidor
    - Dirección IP como número de puerto en el caso de sockets.
  - Enlazar con dicho servidor (verificar si esta sirviendo).
- Estas tareas las realiza el resguardo cliente.
- En el caso de servicios cuya localización no es estándar se recurre al enlace dinámico (dynamic binding).

Enlace Dinámico

- Enlace dinámico
  - Permite localizar objetos con nombre en un sistema distribuido
  - Servidores que ejecutan las RPC.

Enlace Dinámico

- Tipos de enlace:
  - Enlace no persistente
    - La conexión entre el cliente y el servidor se establece en cada llamada RPC.
  - Enlace persistente
    - La conexión se mantiene después de la primera RPC:
    - Útil en aplicaciones con muchas RPC repetidas.
    - Problemas si lo servidores cambian de lugar o fallan.

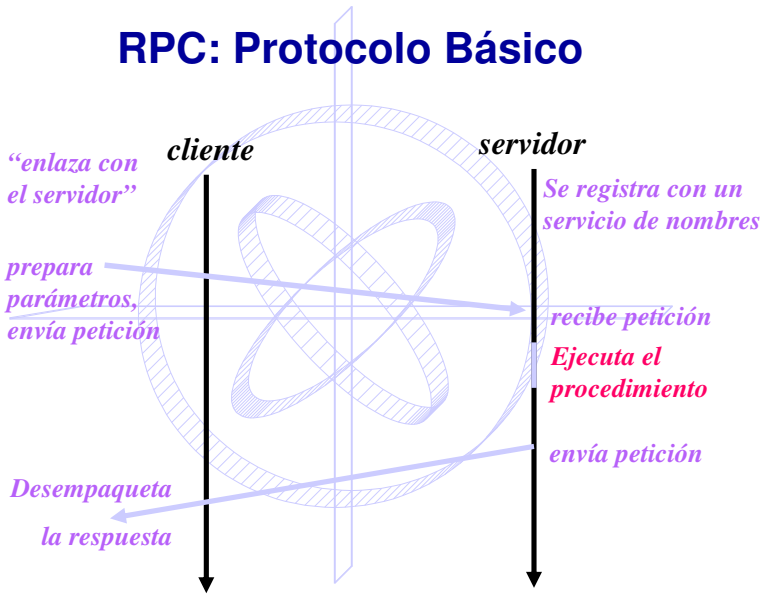
Enlazador Dinámico

- Enlazador dinámico (binder)
  - Es el servicio que mantiene una tabla de traducciones entre nombres de servicio y direcciones
  - Incluye funciones para:
    - Registrar un nombre de servicio (versión).
    - Eliminar un nombre de servicio.
    - Buscar la dirección correspondiente a un nombre de servicio.

Enlazador Dinámico

- Como localizar al enlazador dinámico:
  - Ejecuta en una dirección fija de una computadora fija.
  - El sistema operativo se encarga de indicar su dirección.
  - Difundiendo un mensaje (broadcast) cuando los procesos comienzan su ejecución.

RPC: Protocolo Básico

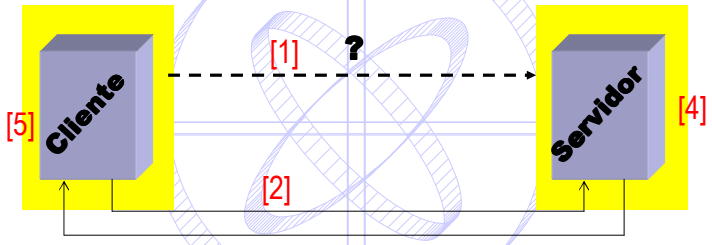


Semántica Fallos

Problemas que pueden plantear las RPC:

- El cliente no es capaz de localizar al servidor. [1]
- Se pierde el mensaje de petición del cliente al servidor. [2]
- Se pierde el mensaje de respuesta del servidor al cliente. [3]
- El servidor falla después de recibir una petición. [4]
- El cliente falla después de enviar una petición. [5]

Semántica Fallos



El cliente no es capaz de localizar al servidor. [1]
Se pierde el mensaje de petición del cliente al servidor. [2]
Se pierde el mensaje de respuesta del servidor al cliente. [3]
El servidor falla después de recibir una petición. [4]
El cliente falla después de enviar una petición. [5]

Cliente no puede localizar al servidor

- El servidor puede estar caído
- El cliente puede estar usando una versión antigua del servidor
- La versión ayuda a detectar accesos a copias obsoletas
- Cómo indicar el error al cliente
  - Devolviendo un código de error (-1)
    - No es transparente
      - Ejemplo: sumar(a,b)
  - Elevando una excepción
    - Necesita un lenguaje que tenga excepciones

Pérdida de Mensajes del Cliente

- Es la más fácil de tratar.
- Se activa una alarma (*timeout*) después de enviar el mensaje.
- Si no se recibe una respuesta se retransmite.
- Depende del protocolo de comunicación subyacente.

Pérdidas de Mensajes de Respuesta

- Más difícil de tratar
- Se pueden emplear alarmas y retransmisiones, pero:
  - ¿Se perdió la petición?
  - ¿Se perdió la respuesta?
  - ¿El servidor va lento?

Pérdidas de Mensajes de Respuesta

- Algunas operaciones pueden repetirse sin problemas (operaciones idempotentes)
- Idempotentes
  - Pueden repetirse n veces sin alterar el resultado
- Una transferencia bancaria no es idempotente
- Solución con operaciones no idempotentes es descartar peticiones ya ejecutadas
  - Un nº de secuencia en el cliente
  - Un campo en el mensaje que indique si es una petición original o una retransmisión

Fallos en los Servidores

- El servidor no ha llegado a ejecutar la operación
  - Se podría retransmitir
- El servidor ha llegado a ejecutar la operación

Fallos en los Servidores

- El cliente no puede distinguir los dos
- ¿Qué hacer?
  - No garantizar nada
  - Semántica al menos una vez
    - Reintentar y garantizar que la RPC se realiza al menos una vez
    - No vale para operaciones no idempotentes
  - Semántica a lo más una vez
    - No reintentar, puede que no se realice ni una sola vez
  - Semántica de exactamente una
    - Sería lo deseable

Fallos en los Clientes

- La computación está activa pero ningún cliente espera los resultados (computación huérfana)
  - Gasto de ciclos de CPU
  - Si el cliente rearranca y ejecuta de nuevo la RPC se pueden crear confusiones

Aspectos de Implementación

- Protocolos RPC
  - Orientados a conexión
    - Fiabilidad se resuelve a bajo nivel, peor rendimiento
  - No orientados a conexión
  - Uso de un protocolo estándar o un específico
    - Algunos utilizan TCP o UDP como protocolos básicos

Aspectos de Implementación

- Coste de copiar información aspecto dominante en rendimiento:
  - buffer del cliente → buffer del SO local → red → buffer del SO remoto + buffer del servidor
  - Puede haber más copias en cliente para añadir cabeceras
  - scatter-gather: puede mejorar rendimiento

RPC de Sun

- Utiliza como lenguaje de definición de interfaz IDL:
  - Una interfaz contiene un nº de programa y un nº de versión.
  - Cada procedimiento específica un nombre y un nº de procedimiento
  - Los procedimientos sólo aceptan un parámetro.
  - Los parámetros de salida se devuelven mediante un único resultado

RPC de Sun

- El lenguaje ofrece una notación para definir:
  - constantes
  - definición de tipos
  - estructuras, uniones
  - programas

RPC de Sun

- rpcgen es el compilador de interfaces que genera:
  - Resguardo del cliente
  - Resguardo del servidor y procedimiento principal del servidor.
  - Procedimientos para el aplanamiento (marshalling)
  - Archivos de cabecera (.h) con los tipos y declaración de prototipos.

RPC de Sun

- Enlace dinámico
  - El cliente debe especificar el host donde ejecuta el servidor
  - El servidor se registra (nº de programa, nº de versión y nº de puerto) en el port mapper local
  - El cliente envía una petición al port mapper del host donde ejecuta el servidor

Ejemplo del IDL (Sun RPC)

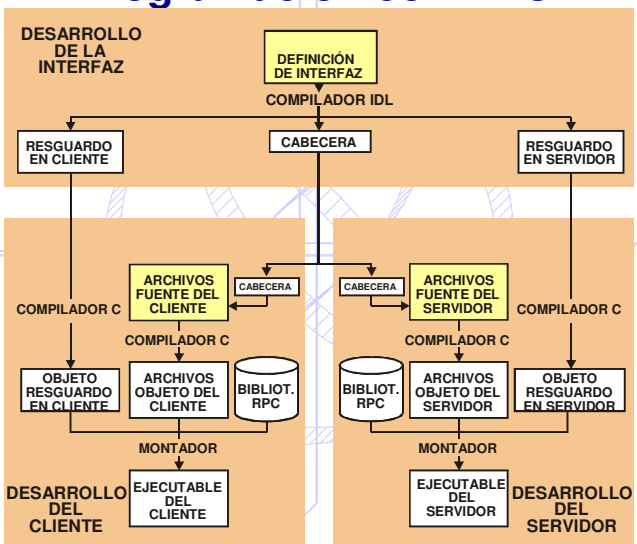
```
struct peticion {
   int a;
   int b;
};

program SUMAR {
   version SUMAVER {
       int SUMA(peticion) = 1;
   } = 1;
} = 99;
```

Programación con un Paquete de RPC

- El programador debe proporcionar:
  - La definición de la interfaz (idl)
    - Nombres de las funciones
    - Parámetros que el cliente pasa al servidor
    - Resultados que devuelve el servidor al cliente
  - El código del cliente
  - El código del servidor
- El compilador de idl proporciona:
  - El resguardo del cliente
  - El resguardo del servidor

Programación con RPC



Ejemplos RPC

- Interfaz IDL (**`ejemplo-rpc/idl`**):
  - Descripción de los servicios:
    - OBTENER_UID
    - OBTENER_NOMBRE
- Código Cliente (**`ejemplo-rpc/cliente`**):
  - Ejecutables que solicitan cada uno de los servicios.
- Código Servidor (**`ejemplo-rpc/servidor`**):
  - Implementación de la interfaz IDL

Esquema de una Aplicación