RPC
Llamadas a procedimientos remotos

Definición

Es una llamada a un procedimiento que existe y se ejecutara en una máquina remota.

Es una técnica para el desarrollo de aplicaciones distribuidas, basadas en el paradigma Cliente/Servidor.

Extiende la noción de llamadas a procedimientos locales, de tal forma que el procedimiento llamado no tiene que estar en la misma máquina donde reside el llamador.

Permite a los programas que llamasen a procedimientos localizados en otras máquinas

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

Operación básica del RPC

Una llamada convencional a un procedimiento, es decir en una sola máquina, funciona de la siguiente manera :

Sea count = read (fd, buf, nbytes); donde:

buf es fd es un entero

un arreglo de caracteres;

nbytes es otro entero.

El programa llamador coloca los parámetros en la pila.

El procedimiento llamado desde el programa llamador se carga en la memoria.

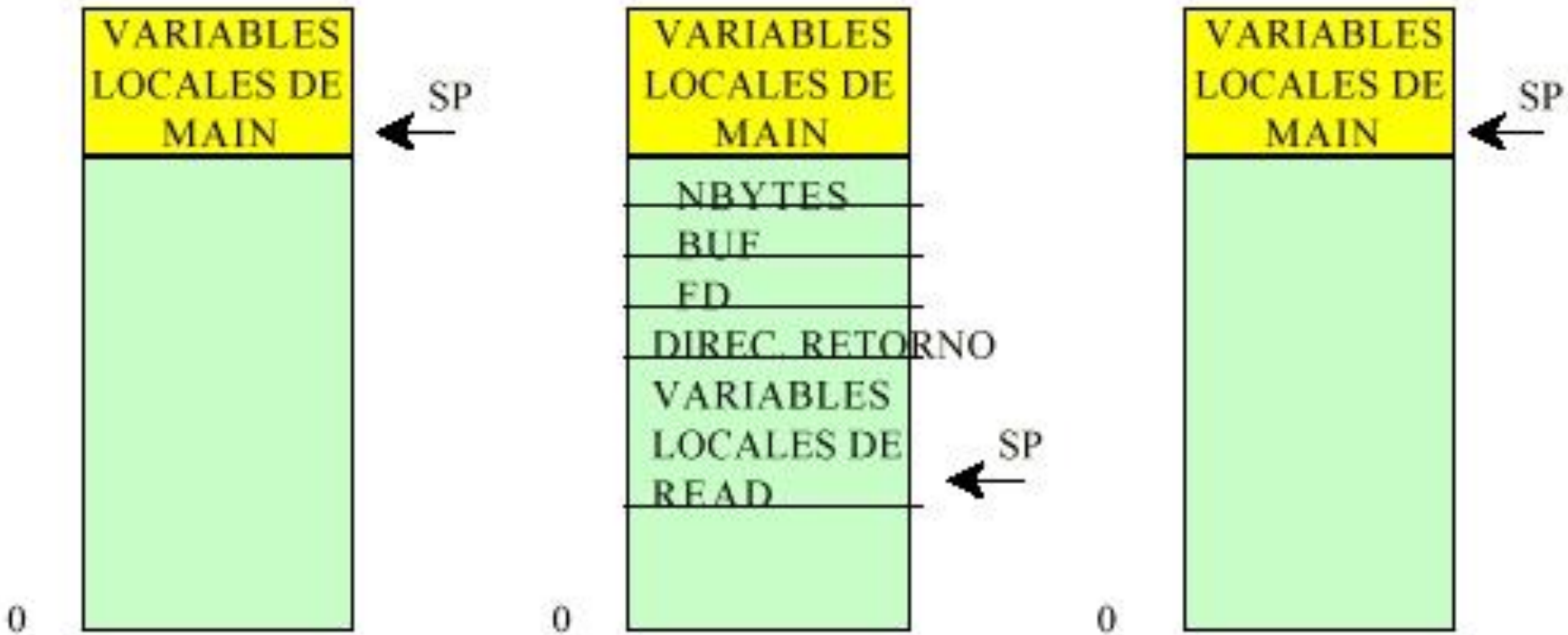
Después de que read termina su ejecución:

Coloca el valor de regreso en un registro.

Elimina la dirección de regreso.

Transfiere de nuevo el control a quien hizo la llamada.

Quien hizo la llamada elimina los parámetros de la pila y regresa a su estado original.



LA PILA ANTES DE LA LLAMADA Aread

LA PILA MIENTRAS EL PROCEDIMIENTO LLAMADO ESTA ACTIVO

LA PILA DESPUES DEL REGRESO A QUIEN HIZO LA LLAMADA

Mecanismos para el paso de parámetros:

Pueden llamarse:

Por valor:

- Se copia a la pila.
- Para el procedimiento que recibe la llamada es solo una variable local ya inicializada.
- El procedimiento podría modificarla, sin que esto afecte el valor de la variable original en el procedimiento que hizo la llamada.

Por referencia:

- Es un apuntador a una variable (es decir, la dirección de la variable), no el valor de la variable.
- En el ej. anterior, válido para “C”, el segundo parámetro es un parámetro por referencia y es un arreglo.
- Si el procedimiento que recibe la llamada utiliza este parámetro por referencia para almacenar algo en el arreglo, modifica el arreglo en el procedimiento que hizo la llamada.

Llamada por copiar / restaurar consiste en:

Quien recibe la llamada copia la variable en la pila, como en la llamada por valor.

La copia de nuevo después de la llamada, escribiendo sobre el valor original.

La decisión de cuál mecanismo utilizar para el paso de parámetros la toman los diseñadores del sistema y es una propiedad fija del lenguaje.

Paso de parámetros

El objetivo de los RPC es que la llamada al procedimiento remoto se parezca lo más posible a una llamada local:

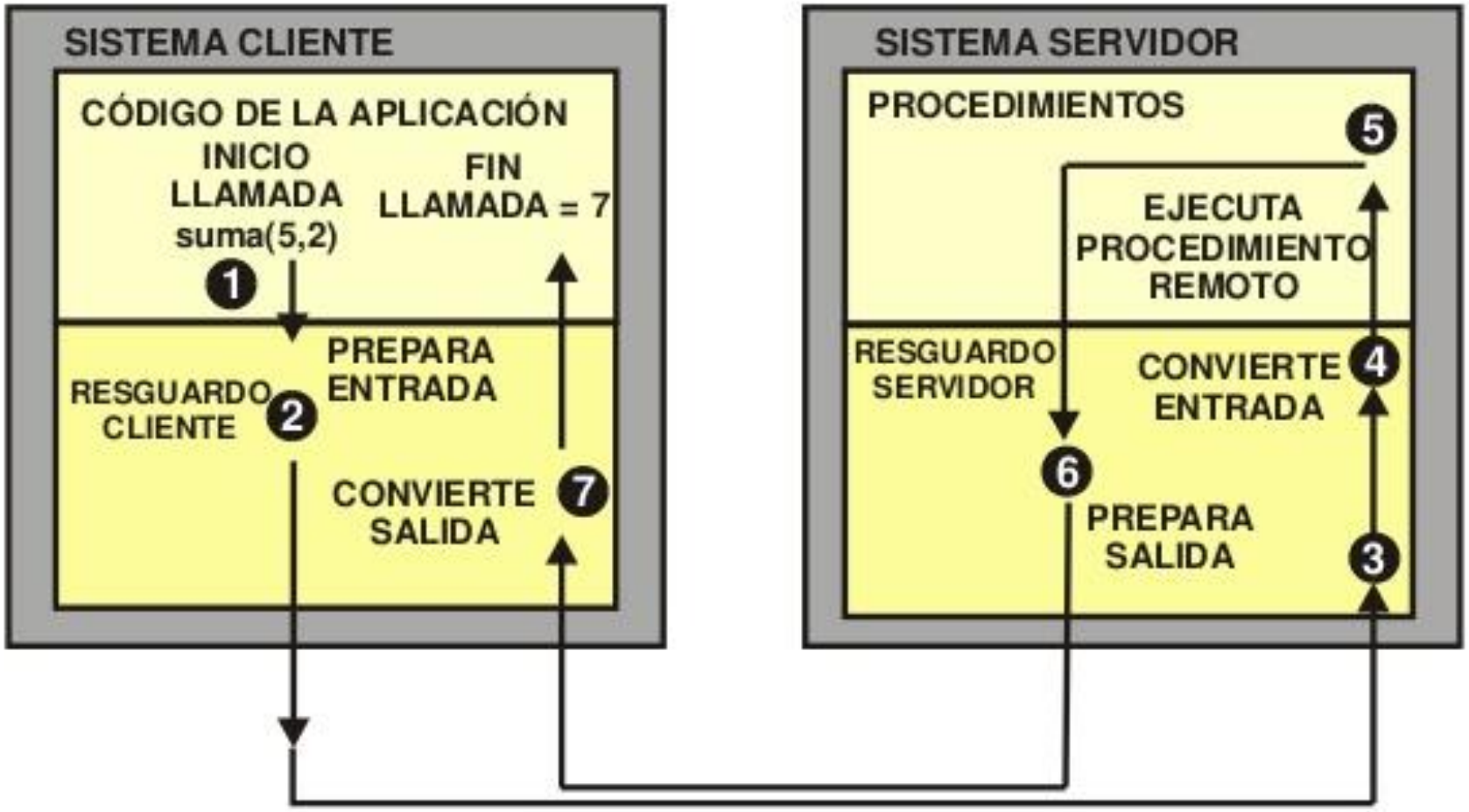
La RPC debe ser transparente.

El procedimiento que hace la llamada no debe ser consciente de que el procedimiento llamado se ejecuta en una máquina distinta, o viceversa.

Si read es un procedimiento remoto (ej.: se ejecuta en la máquina del servidor de archivos) se coloca en la biblioteca una versión distinta de read llamada stub del cliente:

- No coloca los parámetros en registros y le pide al núcleo que le proporcione datos.
- Coloca los parámetros en un mensaje y le pide la núcleo que envíe el mensaje al servidor.
- Después de la llamada a send , el stub del cliente llama a receive y se bloquea hasta que regrese la respuesta.

EL RESGUARDO: Las funciones de abstracción de una llamada RPC a intercambio de mensajes se denominan 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.

FUNCIONES DE RESGUARDOS (stubs):

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

Los procedimientos de resguardo:

En muchos RPC los procedimientos de resguardo se generan en forma automática

Un compilador lee la especificación del servidor y genera un resguardo cliente que empaque sus parámetros en el formato oficial de los mensajes

El compilador también puede generar el resguardo del servidor que los desempaque y que llame al servidor

Cuando el mensaje llega al servidor:

El núcleo lo transfiere a un stub del servidor.

Generalmente el stub del servidor ya habrá llamado a receive y estará bloqueado esperando que le lleguen mensajes.

El stub del servidor: “ desempaca” los parámetros del mensaje.

Llama al procedimiento del servidor de la manera convencional.

Para el servidor es como si tuviera una llamada directa del cliente; lleva a cabo el trabajo y regresa el resultado a quien hizo la llamada, de la forma usual.

El stub del servidor recupera el control luego de la llamada y:

Empaca el resultado en un mensaje.

Llama a send para regresarlo al cliente.

Llama a receive y espera el siguiente mensaje.

Cuando el mensaje regresa a la máquina cliente:

El núcleo ve que está dirigido al proceso cliente.

El mensaje se copia al buffer en espera.

El proceso cliente elimina su bloqueo.

El stub del cliente examina el mensaje, desempaca el resultado, lo copia a quien hizo la llamada y regresa de la manera usual.

Cuando el proceso que hizo la llamada obtiene el control luego de la llamada a read:

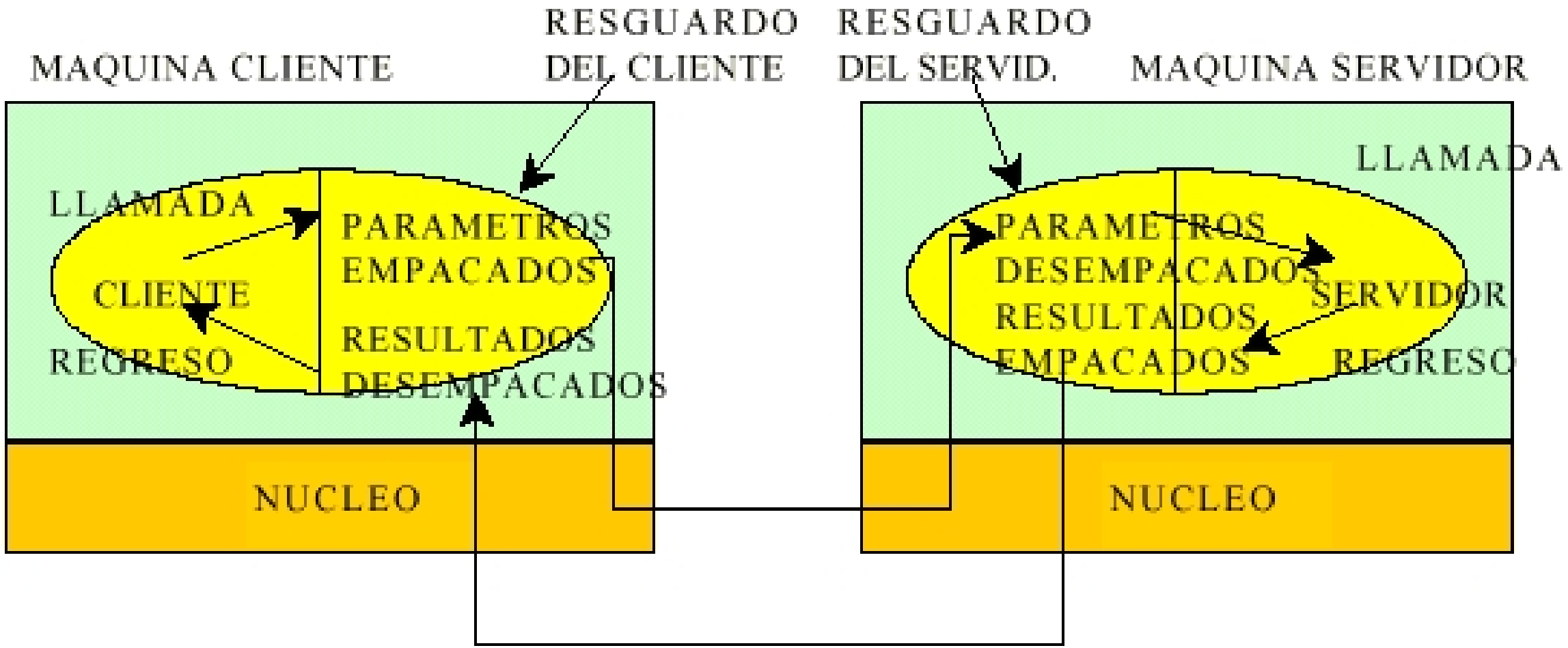
Dispone de los datos.

Ignora que el trabajo se realizó de manera remota.

Ha tenido acceso a servicios remotos mediante llamadas comunes a procedimientos locales .

Se pueden indicar los siguientes pasos como una llamada a un procedimiento remoto:

1. El procedimiento cliente llama al stub del cliente de la manera usual.
2. El stub del cliente construye un mensaje y hace un señalamiento al núcleo.
3. El núcleo envía el mensaje al núcleo remoto.
4. El núcleo remoto proporciona el mensaje al stub del servidor.
5. El stub del servidor desempaca los parámetros y llama al servidor.
6. El servidor realiza el trabajo y regresa el resultado al stub.
7. El stub del servidor empaca el resultado en un mensaje y hace un señalamiento al núcleo.
8. El núcleo remoto envía el mensaje al núcleo del cliente.
9. El núcleo del cliente da el mensaje al stub del cliente.
10. El stub desempaca el resultado y regresa al cliente.



TRANSPORTE DE UN MENSAJE SOBRE LA RED

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).

LOCALIZACION 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).

Cliente no puede localizar al servidor

El servidor podría estar inactivo.

El servidor podría estar utilizando una nueva versión de la interfaz y nuevos resguardos, que no serían compatibles con la interfaz y los resguardos del cliente.

En el servidor, cada uno de los procedimientos regresa un valor:

- Generalmente el código -1 indica un fallo.
- También se suele utilizar una variable global (UNIX) ERRNO a la que se asigna un valor que indica el tipo de error.
- Un tipo de error sería “no se pudo localizar al servidor”.

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.

Si el mensaje realmente se perdió , el servidor no podrá indicar la diferencia entre la retransmisión y el original y todo funcionará bien.

Si el número de mensajes perdidos supera cierto límite , el núcleo puede asumir que el servidor está inactivo y se regresa a la situación “no se pudo localizar al servidor”.

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?

La pérdida de respuestas genera mayores problemas que la pérdida de solicitudes, Se utiliza un cronómetro:

- Si no llega una respuesta en un período razonable, se debe volver a enviar la solicitud.

Pérdidas de Mensajes de Respuesta

Algunas operaciones pueden repetirse sin problemas (operaciones idempotentes)

Idempotentes

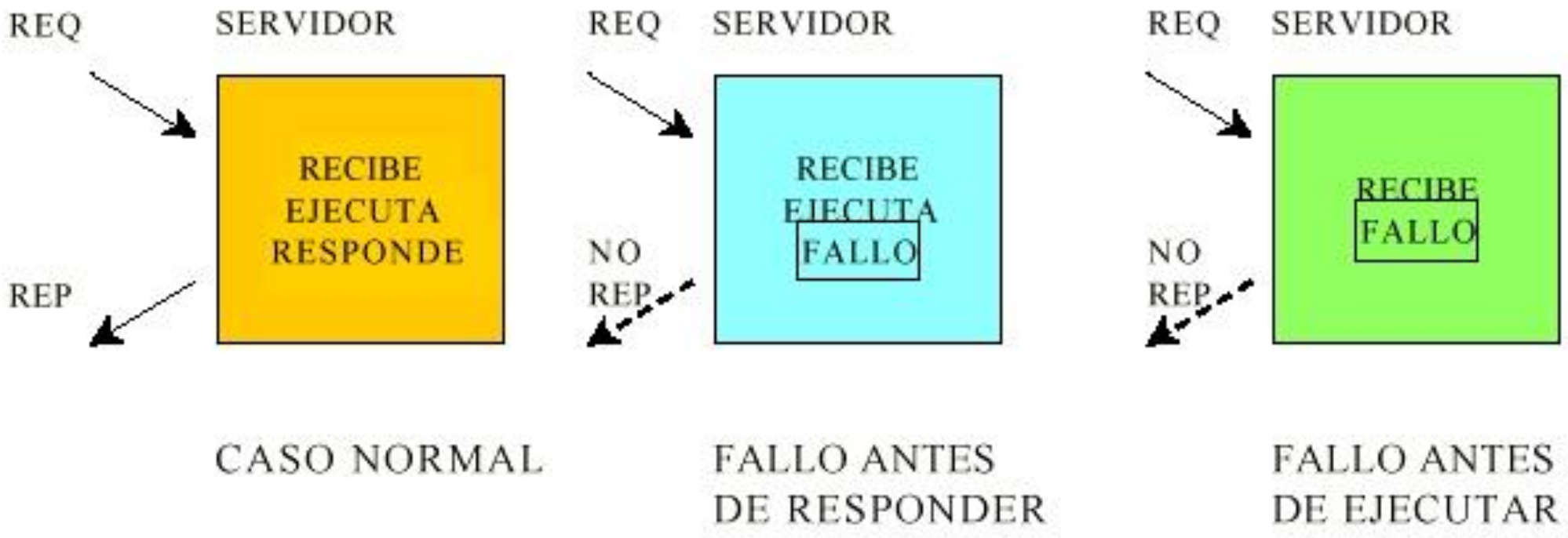
- Pueden repetirse n veces sin alterar el resultado

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 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 una solución deseable

Fallos del Cliente

La cuestión es qué ocurre si un cliente envía una solicitud a un servidor y falla antes de que el servidor responda.

Se genera una solicitud de trabajo o cómputo que al fallar el cliente ya nadie espera; se dice que se tiene un cómputo huérfano.

Un cómputo no deseado se llama huérfano

Los huérfanos pueden provocar varios problemas:

- Desperdiciar ciclos de CPU
- Bloquear archivos o recursos valiosos
- El cliente puede volver a arrancar y llamar de nuevo al RPC, si la respuesta del huérfano llega, puede surgir una confusión

Protocolos RPC

En los protocolos orientados a la conexión

Se establece una conexión entre cliente y servidor.

Todo el tráfico en ambas direcciones utiliza esa conexión.

Se maneja a un nivel inferior mediante el software que soporta la conexión.

Es muy útil para redes de área amplia o extendida (WAN).

Es desventajoso en redes de área local (LAN):

- Por la pérdida de performance que significaría procesar software adicional para aprovechar la ventaja de no perder los paquetes; esto difícilmente se precisa en las LAN.

Muchos sistemas distribuidos en áreas geográficas reducidas utilizan protocolos sin conexión.

En el protocolo estándar de propósito general

La utilización del protocolo estándar IP (o UDP , integrado a IP) posee las siguientes ventajas:

El protocolo ya fue diseñado, lo que ahorra trabajo.

Se dispone de muchas implantaciones, lo que también ahorra trabajo.

Los paquetes IP se pueden enviar y recibir por casi todos los sistemas UNIX.

Los paquetes IP y UDP se pueden transmitir en muchas de las redes existentes.

RPC de Sun

Define:

El formato de los mensajes que el cliente (llamador) envía al invocar un procedimiento remoto.

El formato de los resultados

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

RPCGEN

Permite al programador usar TCP o UDP Usa XDR (external data representation)

Ofrece un compilador que facilita el desarrollo del programa.

Genera código para el cliente y el servidor útil para :

- Empaquetar los argumentos (marshaling)
- Enviar un mensaje RPC Conducir (dipatch) una solicitud del cliente
- Procedimiento adecuado.
  - *Enviar un reply*
  - *Desempaquetar los argumentos*

Ejemplo:

¡Gracias!