5.3 Llamada de procedimiento remoto (RPC)

El concepto de RPC tiene el objetivo de hacer que la programación de los sistemas distribuidos se vea similar a la programación convencional, es decir, lograr un alto nivel de transparencia de distribución. Esto se logra al extender la abstracción de una llamada de procedimiento a entornos distribuidos. Los procedimientos en máquinas remotas pueden llamarse como si fueran procedimientos en el espacio de direcciones locales. El sistema RPC subyacente oculta aspectos importantes de la distribución: la codificación y decodificación de parámetros y resultados, el paso de mensajes y la preservación de la semántica requerida para la llamada al procedimiento.

5.3.1 Problemas de diseño para RPC

Programación con interfaces

Los lenguajes de programación proporcionan un medio para organizar un programa como un conjunto de módulos. La comunicación entre módulos puede realizarse mediante llamadas de procedimiento entre módulos o mediante acceso directo a las variables en otro módulo. Una **interfaz** explícita para cada módulo controla las interacciones entre ellos.

La interfaz de un módulo especifica los procedimientos y las variables a las que se puede acceder desde otros módulos. Los módulos se implementan para ocultar toda la información sobre ellos, excepto la que está disponible a través de su interfaz.

En un programa distribuido, los módulos pueden ejecutarse en procesos separados. En el modelo cliente-servidor cada servidor proporciona un conjunto de procedimientos que están disponibles para el uso de los clientes. Una **interfaz de servicio** es la especificación de los procedimientos ofrecidos por un servidor, definiendo los tipos de argumentos de cada uno de los procedimientos.

- Los programadores solo se preocupan por la abstracción en la interfaz de servicio y no necesitan conocer los detalles de implementación.
- Los programadores tampoco necesitan conocer el lenguaje de programación o la plataforma subyacente utilizada para implementar el servicio (heterogeneidad).
- Las implementaciones pueden cambiar siempre que la interfaz permanezca igual. La interfaz también puede cambiar siempre que siga siendo compatible con la original.

Interfaces de servicio:

- Un módulo cliente que se ejecuta en un proceso no puede acceder a las variables de un módulo en otro proceso. La interfaz de servicio no puede especificar el acceso directo a las variables. En las interfaces CORBA IDL se accede mediante procedimientos getter y setter que se agregan automáticamente a la interfaz.
- El paso de parámetros utilizado en las llamadas a procedimientos locales (llamada por valor y llamada por referencia) no son adecuados entre procesos diferentes. Los parámetros de entrada se pasan al servidor enviando los valores de los argumentos en el mensaje de solicitud y luego como argumentos para la operación dentro del servidor. Los parámetros de salida se devuelven en el mensaje de respuesta y se utilizan como resultado de la llamada o para reemplazar los valores de las variables de la llamada.
  Con un parámetro para entrada y salida, el valor debe transmitirse tanto en los mensajes de solicitud como en los de respuesta.
- Las direcciones no pueden pasarse como argumentos o devolverse como resultados de llamadas a módulos remotos.

Lenguajes de definición de interfaz: Los lenguajes de definición de interfaz (IDL) están diseñados para permitir que los procedimientos implementados en diferentes lenguajes de programación se invoquen entre sí. Proporciona una notación para definir interfaces en las que cada uno de los parámetros de una operación puede describirse como entrada o salida, además de tener su tipo especificado.

La semántica de llamadas asociadas con RPC

Semánticas opcionales: las llamadas a procedimientos remotos pueden o no ejecutarse.

- Fallas de omisión si se pierde el mensaje de solicitud o respuesta.
- Fallas de bloqueo cuando falla el servidor que contiene la operación remota.

Si el mensaje de respuesta no se ha recibido después de un tiempo de espera y no hay reintentos, no está claro si el procedimiento se ha ejecutado. Si se perdió el mensaje de solicitud, el procedimiento no se habrá ejecutado. El procedimiento puede haberse ejecutado y el mensaje resultante se ha perdido. Se puede producir un fallo de bloqueo antes o después de ejecutar el procedimiento. En un sistema asíncrono, el resultado de ejecutar el procedimiento puede llegar después del tiempo de espera. Esta semántica es útil para aplicaciones en las que las llamadas fallidas ocasionales son aceptables.

Semánticas al menos una vez: El invocador recibe un resultado y se sabe que el procedimiento se ejecutó al menos una vez, o recibe una excepción que le informa que no se recibió ningún resultado.Se puede lograr mediante la retransmisión de mensajes de solicitud, que enmascara las fallas de omisión de la solicitud o mensaje de resultado.

- Fallas de bloqueo cuando falla el servidor que contiene la operación remota.
- Cuando el mensaje de solicitud se retransmite, el control remoto el servidor puede recibirlo y ejecutar el procedimiento más de una vez, posiblemente causando valores incorrectos para ser almacenados o devueltos.

Semánticas por lo menos una vez: El invocador recibe un resultado y se sabe que el procedimiento se ejecutó exactamente una vez, o recibe una excepción que le informa que no se recibió ningún resultado, en cuyo caso el procedimiento se habrá ejecutado una vez o nada. Se puede lograr utilizando las siguientes medidas de tolerancia a fallas:

Call semantics

Fault tolerance measures			*Call semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

La transparencia

Todas las llamadas necesarias para los procedimientos de clasificación y transferencia de mensajes estaban ocultas para el programador que realizaba la llamada.

RPC ofrece al menos ***transparencia*** de ubicación y acceso, ocultando la ubicación física del procedimiento y también accediendo a los procedimientos locales y remotos de la misma manera. Middleware también puede ofrecer niveles adicionales de transparencia a RPC.

Las llamadas a procedimientos remotos son más vulnerables a fallas que las locales; involucran una red, otra computadora y otro proceso. Existe la posibilidad de que no se reciba respuesta y, en caso de falla, es imposible distinguir entre la falla de la red y el proceso del servidor. Los clientes que realizan llamadas remotas deben poder recuperarse.

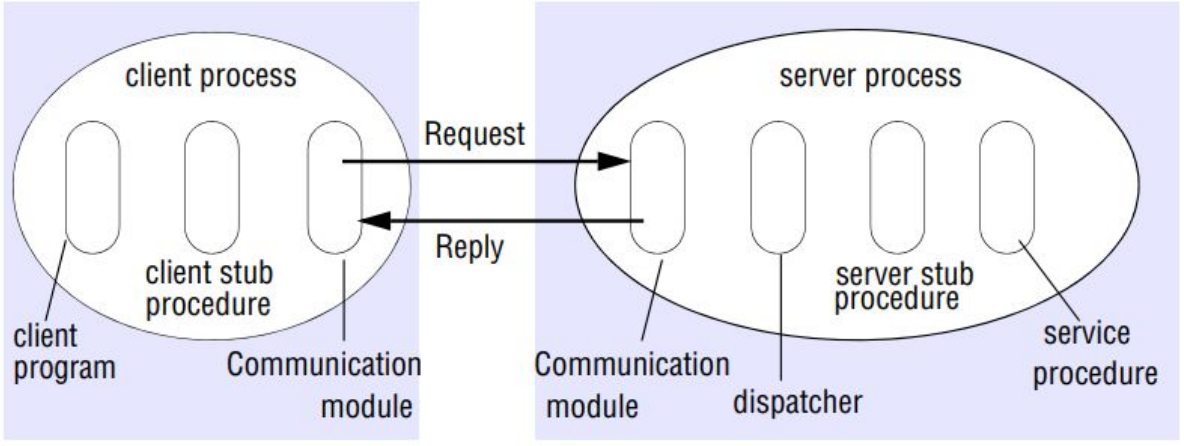
El invocador debe cancelar una llamada a un procedimiento remoto que está tardando demasiado, de tal manera que no tenga ningún efecto en el servidor. El servidor necesitaría poder restaurar las cosas como estaban antes de que se hiciera la llamada. RPC no ofrece llamadas por referencia.

En algunos IDL, una invocación remota puede generar una excepción cuando el cliente no puede comunicarse con un procedimiento remoto. Esto requiere que el programa cliente maneje tales excepciones, permitiéndole lidiar con tales fallas. Un IDL también puede proporcionar una facilidad para especificar la semántica de llamadas de un procedimiento.

Las llamadas remotas deben hacerse transparentes: la sintaxis de una llamada remota es la misma que la de una invocación local, pero la diferencia entre llamadas locales y remotas debe expresarse en sus interfaces.

5.3.2 Implementación de RPC

Role of client and server stub procedures in RPC



El cliente que accede a un servicio incluye un ***procedimiento de código auxiliar (stub procedure)*** para cada procedimiento en la interfaz del servicio. El procedimiento de código auxiliar se comporta como un procedimiento local para el cliente, pero en lugar de ejecutar la llamada, ordena el identificador del procedimiento y los argumentos en un mensaje de solicitud, que envía a través de su módulo de comunicación al servidor. Cuando llega el mensaje de respuesta, desmarca los resultados.

El proceso del servidor contiene un despachador junto con un procedimiento de código auxiliar de servidor y un procedimiento de servicio para cada procedimiento en la interfaz de servicio. El despachador selecciona uno de los procedimientos de código auxiliar del servidor de acuerdo con el identificador del procedimiento en el mensaje de solicitud. El procedimiento de código auxiliar del servidor ordena los argumentos en el mensaje de

solicitud, llama al procedimiento de servicio correspondiente y calcula los valores de retorno para el mensaje de respuesta.

RPC generalmente se implementa a través de un protocolo de solicitud-respuesta. Puede implementarse para tener las semánticas ***al menos una vez*** o ***por lo menos una vez***. El módulo de comunicación implementará las opciones de diseño deseadas en términos de retransmisión de solicitudes, lidiando con duplicados y retransmisión de respuestas.

Preguntas

¿Cuál es el principal objetivo de RPC?
A. Lograr un alto nivel de transparencia de distribución
B. Controlar las interacciones entre los distintos módulos de una aplicación distribuida
C. Manejar las fallas y conservar la integridad de los datos
D. Que la programación de los sistemas distribuidos sea distinta a la convencional

¿Cuales son los 3 problemas de diseño para RPC?
A. Interfaces, transparencia y semántica
B. Sintaxis, manejo de errores e integración de los datos
C. Codificación, decodificación y tiempo de respuesta de parámetros y resultados
D. Retransmisión de solicitudes, mensajes duplicados y retransmisión de respuestas

¿Cuál de los siguientes enunciados es verdadero con respecto a una interfaz de servicio?
A. En CORBA IDL se accede a las variables de otras interfaces mediante getters y setters
B. Los programadores no necesitan conocer los detalles de su implementación
C. Las direcciones pueden pasarse como argumentos o devolverse como resultados.
D. Los parámetros de salida se devuelven en el mensaje de solicitud

¿Cuales son las semánticas de llamadas asociadas con RCP?
A. Maybe, at-least-one, at-most-one
B. No aplicable, retransmisión, reejecución
C. Al-least-one, at-most-one, complete
D. None, maybe, optional

¿Cuales son los componentes de un proceso servidor en la implementación RCP?
A. Módulo de comunicación, despachador, stub procedure y procedimiento de servicio.
B. Petición, respuesta, despachador e invocador
C. Programa, stub procedure y módulo de comunicación
D. Identificador de proceso, mensaje de solicitud, mensaje de respuesta y socket