Resumen unidad IV. Invocacion remota (RPC)

Protocolo Petición Respuesta

Comunicación cliente-servidor. En el caso normal, la comunicación síncrona. Esta comunicación también puede ser fiable. La comunicación asíncrona es una alternativa que puede ser útil en situaciones donde los clientes pueden recuperar las respuestas más tarde.

Se basa en tres primitivas de comunicación:

- *doOperation*(s,args): invoca una operación remota en s, con los argumentos indicados en args (parámetros y tipo de operación).
- *getRequest*: espera/adquiere una petición en el servidor.
- *sendReply*(r,c): manda el mensaje de respuesta r al cliente c.

Modelo de Fallos

Si las tres primitivas se implementan con datagramas UDP, adolecerán de los mismos fallos de comunicación que cualquier otro ejemplo de aplicación de UDP (Fallos por omisión, duplicacion o desorden de mensajes, fallo de los procesos).

Solucion Fallo por Omisión: timeout en doOperation().

- Opción 1: la operación doOperation simplemente retorne un indicador de fallo.
- Opción 2: reintentar doOperacion de forma repetida hasta se obtenga una respuesta.

Solucion de Fallo Bizantino:

- Opción 1: Puede implementar un mecanismo para descartarlas si todavía está realizando la operación. Esto es posible con el identificador de petición:

```
id_cliente # id_peticion
```

- Opción 2: Si ya la ha realizado y la operación es idempotente, la realiza de nuevo. Si no, puede implementar un histórico con los resultados de las operaciones realizadas.

Estilos de protocolos RPC

En la implementación de los distintos tipos de RPC se utilizan tres protocolos con diferentes semánticas en presencia de fallos de comunicación:

- El protocolo petición (R): cuando el procedimiento no devuelve ningún valor y el cliente no necesita confirmación de que ha sido ejecutado.
- El protocolo petición-respuesta (RR): los mensajes de respuesta del servidor sirven como confirmación de las peticiones cliente.
- El protocolo petición-respuesta-confirmación de la respuesta (RRA): está basado en el intercambio de tres mensajes: petición-respuesta-confirmación.

`Http`: basado en el protocolo RR, con un conjunto fijo de metodos (GET, POST, PUT, etc.). Implementado sobre TCP. Permite negociacion y autenticacion

Ejs. de metodos Http:

- GET: pide recursos al servidor con el url proporcionado.
- POST: generalmente para indicar la creacion de recursos.
- PUT: similar a POST, pero por convencion se utiliza para indicar la actualizacion de un recurso.
- DELETE: para indicar la eliminacion de datos.
- OPTIONS: el servidor proporciona al cliente una lista de métodos aplicables a un URL (GET, HEAD, PUT, etc.) y sus requisitos.
- TRACE: el servidor envía de vuelta el mensaje de petición. Se utiliza en procesos de depuración.

Códigos de respuesta HTTP:

1. Respuestas informativas (100–199)
2. Respuestas satisfactorias (200–299)
3. Redirecciones (300–399)
4. Errores de los clientes (400–499)
5. Errores de los servidores (500–599)

RPC

La llamada a procedimiento remoto representa un adelanto en computación distribuida, haciendola casi idéntica a la programación convencional. Logra un alto nivel de transparencia de distribución. Esta unificación se logra extendiendo la abstracción de una llamada de procedimiento a entornos distribuidos.

Los servidores pueden ser clientes de otros servidores para permitir cadenas de RPC. Un proceso servidor define en su interfaz de servicio los procedimientos disponibles para ser llamados remotamente. RPC se implementa usualmente sobre un protocolo petición-respuesta.

El cliente que accede a un servicio incluye un procedimiento de resguardo para cada procedimiento en la interfaz de servicio. Procedimiento de resguardo es similar al de un proxy. Se comporta como un procedimiento local del cliente, pero en lugar de ejecutar la llamada, empaqueta el identificador del procedimiento y los argumentos en un mensaje de petición, que se envía vía su módulo de comunicación al servidor.

Cuando llega el mensaje de respuesta, desempaqueta los resultados. El proceso servidor contiene un distribuidor junto a un procedimiento de resguardo de servidor y un procedimiento de servicio para cada procedimiento de la interfaz

El distribuidor selecciona uno de los procedimientos de resguardo según el identificador de procedimiento del mensaje de petición. Se llama al procedimiento de servicio correspondiente y se empaquetan los datos con el resultado para el mensaje de respuesta.

Los procedimientos de servicio implementan los procedimientos en la interfaz
de servicio

Interfaces

La mayoría de los Lenguajes proporcionan medios para que varios módulos puedan comunicarse entre sí: las interfaces.
No es posible para un módulo que se ejecuta en un proceso acceder a las variables de un módulo que está en otro proceso.

Parámetros de entrada:

- Se pasan al módulo remoto mediante el envío de los valores de los argumentos en el mensaje de petición.
- Posteriormente se proporcionan como argumentos a la operación que se ejecutará en el servidor.

Parámetros de salida:

- Se devuelven en el mensaje de respuesta y se sitúan como la respuesta de la llamada o reemplazando valores del argumento en el entorno.
- Los punteros en un proceso dejan de ser válidos en el remoto. En consecuencia, no pueden pasarse punteros como argumentos o como valores retornados.
- *Interfaces de servicio*: En el modelo cliente-servidor, cada servidor proporciona procedimientos disponibles para clientes. El término se emplea para referirse a la especificación de los procedimientos que ofrece un servidor. Por ejemplo, un servidor de archivos proporcionará procedimientos para leer y escribir archivos
- *Interfaces remotas*: En el modelo de objetos distribuidos, una interfaz remota especifica los métodos de un objeto que están disponibles para su invocación por objetos de otros procesos, y define los tipos de los argumentos de entrada y de salida. La gran diferencia es que los métodos en las interfaces remotas pueden pasar objetos como argumentos y como resultados de los métodos.

Los lenguajes de definición de interfaces (IDL):
Permiten que los objetos implementados en lenguajes diferentes se invoquen unos a otros. Proporciona notación para definir interfaces en la cual cada uno de los parámetros de un método se podrá describir como de entrada o de salida además de su especificación de tipo.

Semánticas de invocación:

- *"Pudiera ser"*: el que invoca no puede decir si un metodo se ejecuto una o ninguna vez (ninguna medida de tolerancia a fallos).
- *"Al menos una vez"*: el invocante recbie un resultado, por ende sabe que se ejecuto el metodo aunque sea una vez (salvo que reciba una exepcion como respuesta). Se logra con retrtansminsion de mensajes de peticion. Puede sufrir fallos de caida del servidor o que se ejecute mas de una vez el metodo (causando errores).
- *"Como maximo una vez"*: el invocante recibe bien un resultado, o una excepción que le informa de que no se recibió el resultado, de modo que el método se habrá ejecutado o una vez o ninguna en absoluto. Ejemplos son CORBA y Java RMI.

Modelo de objetos distribuido

En RMI, un objeto llamante puede invocar un método en un objeto potencialmente remoto.

- Invocaciones de métodos remotas: Las invocaciones de métodos entre objetos en diferentes procesos.

- Invocaciones de métodos locales: Las invocaciones de métodos entre objetos del mismo proceso.

Objetos Remotos: Objetos que pueden recibir invocaciones remotas.

Los *2* conceptos fundamentales siguientes son el corazón del modelo objetos distribuidos:

- Referencia de objeto remoto: otros objetos pueden invocar los métodos de un objeto remoto si tienen acceso a su “referencia de objeto remoto”.
- Interfaz remota: cada objeto remoto tiene una interfaz remota que especifica cuáles de sus métodos pueden invocarse remotamente.

Referencias a objetos remotos: permite que cualquier objeto que pueda recibir un RMI tenga una referencia a objeto remoto (un identificador para referirse a un objeto remoto particular único).

Las referencias a objetos remotos son análogas a las locales en cuanto a que el objeto remoto donde se recibe la invocación de método remoto se especifica mediante una referencia a objeto remoto. Las referencias a objetos remotos pueden pasarse como argumentos y resultados de las invocaciones de métodos remotos.

Interfaces remotas: La clase de un objeto remoto implementa los métodos de su interfaz remota. Los objetos en otros procesos pueden invocar solamente los métodos que pertenezcan a su interfaz remota.

Acciones en un sistema de objetos distribuido: Una acción se inicia mediante la invocación de un método pero los objetos involucrados pueden estar en computadores o procesos distintos.

Excepciones: Cualquier invocación remota puede fallar. Es así, que una invocación a un método remoto debiera ser capaz de lanzar excepciones tales como timeouts.

Implementación de RMI

Módulo de comunicación: Los dos módulos cooperantes realizan el protocolo de petición-respuesta, que retransmite los mensajes de petición y respuesta entre el cliente y el servidor.

El *módulo de comunicación* emplea sólo los tres primeros elementos, que especifican: el tipo de mensaje, su idPeticion y la referencia remota del objeto que se invoca. El idMetodo y todo el empaquetado y desempaquetado es cuestión del software RMI.

Los módulos de comunicación son responsables conjuntamente de proporcionar una semántica de invocación. El módulo de comunicación en el servidor selecciona el distribuidor para la clase del objeto que se invoca, pasando su referencia local, que se obtiene del módulo de referencia remota en respuesta al identificador de objeto remoto en el mensaje

El *modulo de referencia remota*: Traduce las referencias enttre objetos locales y remotots y crea referencias a obnjetos remotos. Los componentes del módulo software de RMI llaman a este módulo cuando realizan el empaquetado y el desempaquetado de las referencias a objetos remotos.

Cada modulo tiene una tabla que contiene:

- Una entrtada por cada objeto remoto
- Una entrada por cada proxy

Las acciones del módulo de referencia remota ocurren como sigue:

1. Cuando se pasa un objeto remoto por primera vez, se le pide al módulo de referencia remota que cree una referencia a un objeto remoto, que se añade a su tabla.
2. Cuando llega una referencia a un objeto remoto, en un mensaje de petición o respuesta, se le pide al módulo de referencia remota la referencia al objeto local correspondiente. En el caso de que el objeto remoto no esté en la tabla, el software RMI crea un nuevo proxy y pide al módulo de referencia remota que lo añada a la tabla.

El *software de RMI* consiste en una capa de software entre los objetos del nivel de aplicación y los módulos de comunicación y de referencia remota.

Los papeles de los objetos de middleware son como sigue:

- *Proxy*: hace que la invocación al método remoto sea transparente para los clientes. Para ello se comporta como un objeto local para el que invoca; pero en lugar de ejecutar la invocación, dirige el mensaje al objeto remoto.
- Hay un proxy para cada objeto remoto del que el cliente disponga de una referencia de objeto remoto. La clase de un proxy implementa los métodos de la interfaz remota del objeto remoto al que representa. Esto asegura que las invocaciones al método remoto son adecuadas según el tipo del objeto remoto.
- Cada Servidor tiene un Distribuidor y un Esqueleto para cada clase que represente a un objeto remoto.

Distribuidor: El distribuidor recibe el mensaje de petición desde el módulo de comunicación. Emplea el idMetodo para seleccionar el método apropiado del esqueleto, pasándole el mensaje de petición. El distribuidor y el proxy emplean los mismos métodos de asignación de cada idMetodo para los métodos de la interfaz remota.

Esqueleto: la clase de un objeto remoto tiene un esqueleto, que implementa los métodos interfaz remota. Se encuentran implementados de forma muy diferente de los métodos del objeto remoto. Un método del esqueleto desempaqueta los argumentos del mensaje de petición, invoca el método correspondiente en el objeto remoto.

Cuando un cliente invoca un método en un objeto remoto, se envía un mensaje de invocación al proceso servidor que alberga el objeto remoto. Este mensaje necesita especificar el objeto particular cuyo método se va a
invocar.

Una referencia a un objeto remoto es un identificador para un objeto remoto que es válida a lo largo y ancho de un sistema distribuido. En el mensaje de invocación se incluye una referencia a objeto remoto que especifica cuál es el objeto invocado.