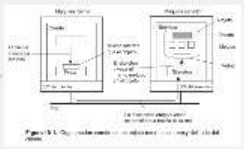
Sistemas Basados en Objetos Distribuidos

Arquitectura

En popular símbolo a la habilidad natural de construir software en la forma de componentes independientes más o menos bien definidos.

Objetos Distribuidos
La característica fundamental de un objeto es que encapsula datos, llamado el estado, y las operaciones incluidas en tales datos, llamadas métodos. Los métodos se ponen a disposición mediante una interfaz



En un objeto distribuido general, el estado puede estar físicamente distribuido en varias máquinas, pero su distribución también está oculta de los clientes detrás de sus interfaces.

Cuando un cliente se vincula a un objeto distribuido, una implementación de la interfaz de objeto, llamada **proxy**, se carga entonces en el espacio direcciones y la dirección del cliente.

El objeto real reside en la máquina de un servidor, donde ofrece la misma interfaz que en la máquina del cliente.
Las solicitudes de invocación entrantes primero se pasan al resguardo de un servidor, el cual las desempaqueta para crear invocaciones a métodos en la interfaz del objeto ubicada en el servidor.

El resguardo del lado del cliente a menudo se comoca como plantilla de código (**skeleton**) ya que proporciona los medios básicos para permitir que el middleware del servidor acceda a los objetos definidos por el usuario.

Una característica, aunque un tanto contraintuitiva de la mayoría de los objetos distribuidos es que **su estado no es distribuido**: reside en una sola máquina.

Solamente las interfaces implementadas por el objeto están disponibles en otras máquinas. Esos objetos también se conocen como **objetos remotos**.

Tiempo de compilación versus objetos en tiempo de ejecución

En los sistemas distribuidos, los objetos aparecen en muchas formas. La forma que resulta más evidente es la directamente relacionada con objetos a nivel de lenguaje, tales como aquellos sopor tados por Java, C++, u otros lenguajes, los cuales se conocen como **objetos en tiempo de compilación**. En este caso, un objeto se define como **la instancia de una clase.**

La desventaja evidente de los objetos en tiempo de compilación es la dependencia de un lenguaje de programación. Por consiguiente, una forma alternativa de construir objetos distribuidos es **hacerlo explícitamente durante el tiempo de ejecución.**

Construir objetos distribuidos en tiempo de ejecución es independiente del lenguaje de programación en el cual están escritas las aplicaciones distribuidas. En particular, se puede construir una aplicación a partir de objetos escritos en varios lenguajes.

Cuando se trata con objetos en tiempo de ejecución, el cómo se implementan es realidad básicamente se deja abierto.

Lo esencial es cómo hacer que esa implementación parezca ser un objeto sopre métodos puedan ser invocados desde una máquina remota.

Un método común es utilizar un adaptador de objeto, el cual actúa como envolvente alrededor de la implementación con el único propósito de darle la apariencia de un objeto.

Para hacer la envoltura tan fácil como sea posible, los objetos se definen únicamente en función de las interfaces que implementan. La implementación de una interfaz puede entonces ser registrada en un adaptador, el cual posteriormente hace que la interfaz sea disponible para invocaciones (remotas).

El adaptador se encargará de que las solicitudes de invocación sean atendidas y, por tanto, de proporcionar una imagen de objetos remotos a sus clientes.

Objetos persistentes y transitorios

Además de la distinción entre objetos a nivel de lenguaje y objetos en tiempo de ejecución, también existe una distinción entre objetos persistentes y objetos transitorios.

Un objeto persistente es uno que continúa existiendo aun cuando ya no está contenido en el espacio de dirección de cualquier proceso de servidor.

En otros términos, un objeto persistente no depende de su servidor. En la práctica, esto significa que el servidor que actualmente está manejando el objeto persistente puede guardar su estado en un almacenamiento secundario y luego salir.

Un objeto transitorio es un objeto que existe sólo en tanto el servidor que lo está alojando exista.
En cuanto el servidor deja de funcionar, el objeto deja de existir.

Comunicación

Sistemas distribuidos basados en objetos, ofrecen medios para que los clientes invoquen un objeto.
Basada en llamadas a procedimientos remotos (RPC)

Vinculación de un cliente a un objeto

Las referencias a objetos pueden ser pasadas libremente entre los procesos en diferentes máquinas, por ejemplo, como parámetros ante invocaciones a métodos.

Proporcionan referencias a objetos a nivel de todo el sistema

Implementación de referencias a un objeto

Una referencia simple a un objeto incluirá la dirección de red de la máquina donde reside el objeto, junto con un punto final que identifica al servidor que maneja el objeto, más una indicación acerca de qué objeto se trata.

Servidor de localización

expande la idea de un servidor de nombres local que mantiene una tabla de puntos de terminación para un servidor de localización que se puede ubicar en cada máquina donde el servidor de un objeto está funcionando actualmente.

Control de implementación

Este método puede incluir lineas de un poco más sofisticado a medio de control de implementación en la referencia a un objeto, el cual se refiere a una implementación adecuada de un objeto para que el cliente pueda cargar dinámicamente cuando se vincula al objeto.

Vinculación explícita

Con vinculación implícita, al cliente se le ofrece un mecanismo simple que le permite invocar directamente métodos que utilizan sólo una referencia a un objeto.
El cliente está ligado transparentemente al objeto en el momento en que la referencia se transforma en el objeto real

Vinculación implícita

el cliente primero deberá invocar una función especial para ligarse al objeto antes de que en realidad pueda invocar sus métodos. La vinculación explícita, en general, regresa un apuntador a un proxy que luego llega a estar localmente disponible

Paso de parámetros.

Sin embargo, existen algunas sutilezas que pueden hacer que las RMI sean más engañosas de lo que inicialmente se podría esperar, tal como analizamos brevemente en las páginas siguientes.

En otros términos, todos los objetos presentes en el sistema pueden ser accesados desde máquinas remotas. En ese caso, podemos utilizar consistentemente las referencias a objetos como parámetros en las invocaciones a métodos.

Cuando se invoca un método con la referencia a un objeto como parámetro, dicha referencia se copia y pasa como un parámetro de valor sólo cuando se refiere a un objeto remoto. En este caso, el objeto es literalmente pasado mediante referencia. Sin embargo, cuando la referencia se refiere a un objeto local, es decir un objeto situado en el mismo espacio de dirección que el cliente, el objeto referido se copia en su totalidad y pasa junto con la invocación. En otros términos, el objeto se pasa por valor

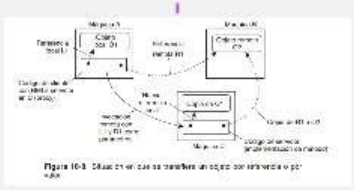


Figura 10-3. Situación en la que se transfiere un objeto con referencia o por valor

Invocaciones a métodos remotos estáticas versus dinámicas

Una diferencia esencial entre RMI y RPC es que las RMI, en general, soportan referencias a un objeto entre todo el sistema como ya explicamos.

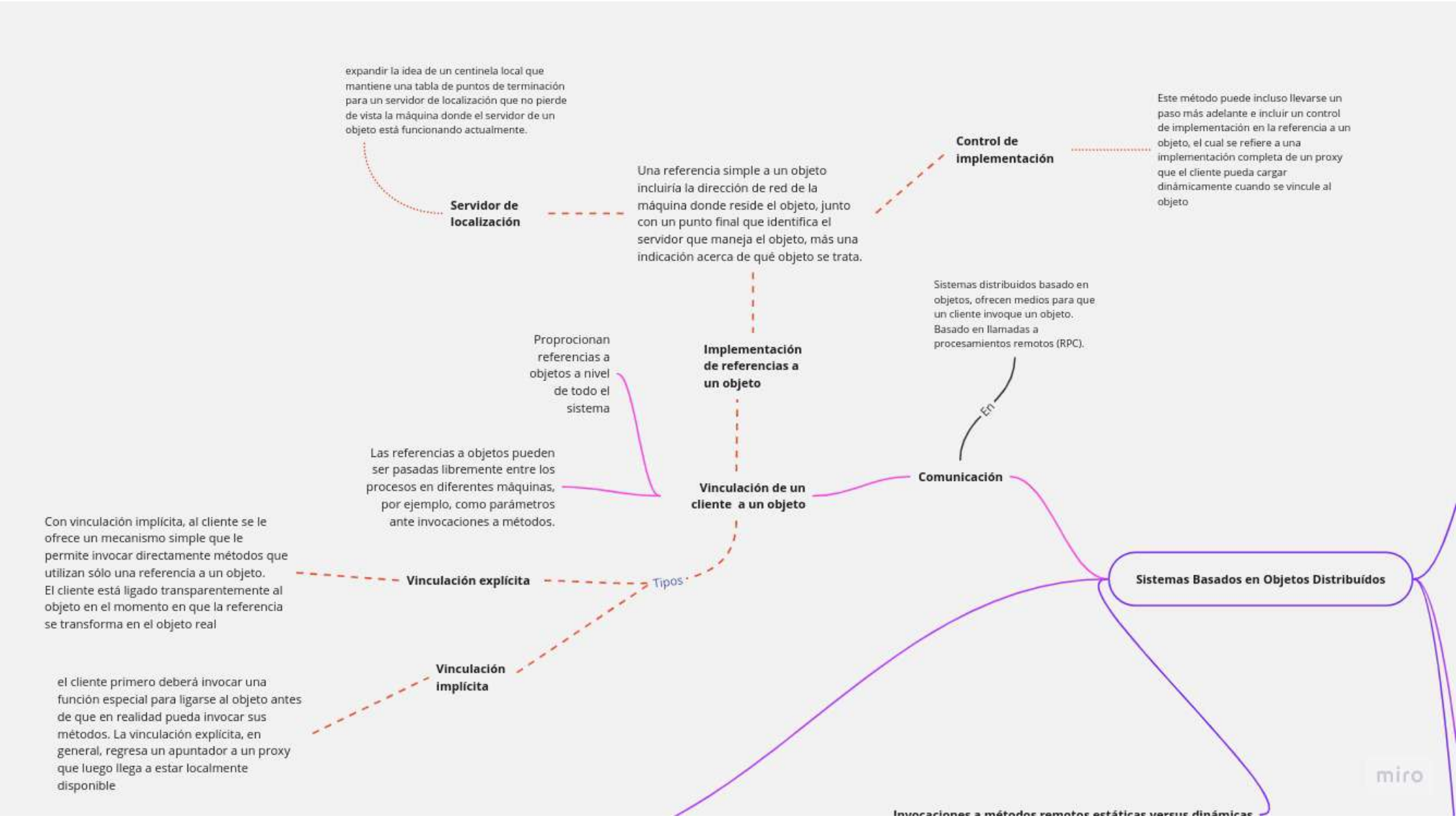
Método de utilizar definiciones de interfaz predefinidos se conoce, en general, como invocación estática. Las invocaciones estáticas requieren que se conozcan las interfaces de un objeto cuando la aplicación del cliente se está desarrollando.

En particular, en ocasiones resulta conveniente ser capaces de componer una invocación a un método en tiempo de ejecución, esto es conocido también como invocación dinámica.

La invocación dinámica adopta, en general, una forma como la siguiente:

invoke(object_method, input_parameters, output_parameters);

Donde object identifica el objeto distribuido, method es un parámetro que especifica con exactitud qué método deberá ser invocado. Input_parameters es una estructura de datos que retiene los valores de los parámetros de entrada del método, y output_parameters se refiere a una estructura de datos donde los valores de salida pueden ser almacenados.


Sistemas Basados en Objetos Distribuídos
Comunicación
En
Sistemas distribuidos basado en objetos, ofrecen medios para que un cliente invoque un objeto. Basado en llamadas a procesamientos remotos (RPC).
Vinculación de un cliente a un objeto
Tipos
Vinculación explícita
Con vinculación implícita, al cliente se le ofrece un mecanismo simple que le permite invocar directamente métodos que utilizan sólo una referencia a un objeto. El cliente está ligado transparentemente al objeto en el momento en que la referencia se transforma en el objeto real
Vinculación implícita
el cliente primero deberá invocar una función especial para ligarse al objeto antes de que en realidad pueda invocar sus métodos. La vinculación explícita, en general, regresa un apuntador a un proxy que luego llega a estar localmente disponible
Las referencias a objetos pueden ser pasadas libremente entre los procesos en diferentes máquinas, por ejemplo, como parámetros ante invocaciones a métodos.
Proprocionan referencias a objetos a nivel de todo el sistema
Implementación de referencias a un objeto
Una referencia simple a un objeto incluiría la dirección de red de la máquina donde reside el objeto, junto con un punto final que identifica el servidor que maneja el objeto, más una indicación acerca de qué objeto se trata.
Servidor de localización
expandir la idea de un centinela local que mantiene una tabla de puntos de terminación para un servidor de localización que no pierde de vista la máquina donde el servidor de un objeto está funcionando actualmente.
Control de implementación
Este método puede incluso llevarse un paso más adelante e incluir un control de implementación en la referencia a un objeto, el cual se refiere a una implementación completa de un proxy que el cliente pueda cargar dinámicamente cuando se vincule al objeto
Invocaciones a métodos remotos estáticas versus dinámicas
miro

Invocaciones a métodos remotos estáticas versus dinámicas

Paso de parámetros.

Una diferencia esencial entre RMI y RPC es que las RMI, en general, soportan referencias a un objeto entre todo el sistema como ya explicamos.

Sin embargo, existen algunas sutilezas que pueden hacer que las RMI sean más engañosas de lo que inicialmente se podría esperar, tal como analizamos brevemente en las páginas siguientes.

Método de utilizar definiciones de interfaz predefinidas se conoce, en general, como invocación estática. Las invocaciones estáticas requieren que se conozcan las interfaces de un objeto cuando la aplicación del cliente se está desarrollando.

En otros términos, todos los objetos presentes en el sistema pueden ser accesados desde máquinas remotas. En ese caso, podemos utilizar consistentemente las referencias a objetos como parámetros en las invocaciones a métodos.

La invocación dinámica adopta, en general, una forma como la siguiente:

invoke(object_method, input_parameters, output_parameters);

En particular, en ocasiones resulta conveniente ser capaces de componer una invocación a un método en tiempo de ejecución, esto es conocido también como invocación dinámica.

Cuando se invoca un método con la referencia a un objeto como parámetro, dicha referencia se copia y pasa como un parámetro de valor sólo cuando se refiere a un objeto remoto. En este caso, el objeto es literalmente pasado mediante referencia. Sin embargo, cuando la referencia se refiere a un objeto local, es decir un objeto situado en el mismo espacio de dirección que el cliente, el objeto referido se copia en su totalidad y pasa junto con la invocación. En otros términos, el objeto se pasa por valor

Donde object identifica el objeto distribuido, method es un parámetro que especifica con exactitud qué método deberá ser invocado, *input_parameters* es una estructura de datos que retiene los valores de los parámetros de entrada del método, y *output_parameters* se refiere a una estructura de datos donde los valores de salida pueden ser almacenados.

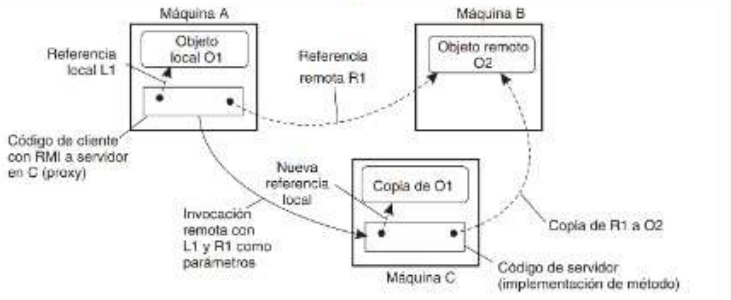


Figura 10-8. Situación en que se transfiere un objeto por referencia o por valor.

Este método puede incluso llevarse un paso más adelante e incluir un control de implementación en la referencia a un objeto, el cual se refiere a una implementación completa de un proxy que el cliente pueda cargar dinámicamente cuando se vincule al objeto

Sistemas Basados en Objetos Distribuídos

Arquitectura

Es popular debido a la habilidad natural de construir software en la forma de componentes independientes más o menos bien definidos.

Objetos Distribuídos

La característica fundamental de un objeto es que encapsula datos, llamado el estado, y las operaciones incluidas en tales datos, llamadas métodos.
Los métodos se ponen a disposición mediante una interfaz

Cuando un cliente se vincula a un objeto distribuido, una implementación de la interfaz de objeto, llamada **proxy**, se carga entonces en el espacio destinado a la dirección del cliente.

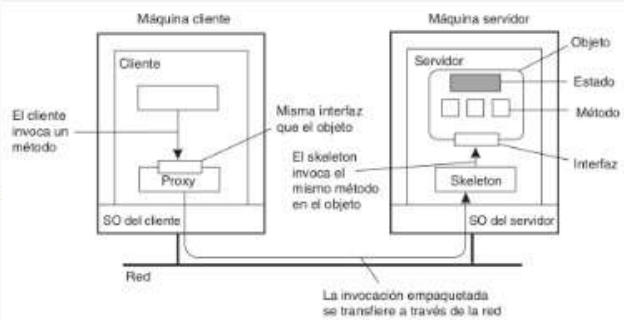


Figura 10-1. Organización común de un objeto remoto con proxy del lado del cliente.

En un objeto distribuido general, el estado puede estar físicamente distribuido en varias máquinas, pero su distribución también está oculta de los clientes detrás de sus interfaces.

El objeto real reside en la máquina de un servidor, donde ofrece la misma interfaz que en la máquina del cliente.
Las solicitudes de invocación entrantes primero se pasan al resguardo de un servidor, el cual las desempaqueta para crear invocaciones a métodos en la interfaz del objeto ubicada en el servidor.

El resguardo del lado del cliente a menudo se conoce como plantilla de código (**skeleton**) ya que proporciona los medios básicos para permitir que el middleware del servidor acceda a los objetos definidos por el usuario.

Una característica, aunque un tanto contraintuitiva de la mayoría de los objetos distribuidos es que **su estado no es distribuido**: reside en una sola máquina.

Solamente las interfaces implemen tadas por el objeto están disponibles en otras máquinas. Esos objetos también se conocen como **objetos remotos**.

státicas versus dinámicas

conocen como **objetos remotos**.

Tiempo de compilación versus objetos en tiempo de ejecución

En los sistemas distribuidos, los objetos aparecen en muchas formas. La forma que resulta más evidente es la directamente relacionada con objetos a nivel de lenguaje, tales como aquellos sopor tados por Java, C++, u otros lenguajes, los cuales se conocen como **objetos en tiempo de compilación**. En este caso, un objeto se define como **la instancia de una clase.**

La desventaja evidente de los objetos en tiempo de compilación es la dependencia de un lenguaje de programación. Por consiguiente, una forma alternativa de construir objetos distribuidos es **hacerlo explícitamente durante el tiempo de ejecución.**

Construir objetos distribuídos en tiempo de ejecución es independiente del lenguaje de programación en el cual están escritas las aplicaciones distribuidas. En particular, se puede construir una aplicación a partir de objetos escritos en varios lenguajes.

Cuando se trata con objetos en tiempo de ejecución, el cómo se implementan en realidad básicamente se deja abierto.

Lo esencial es cómo hacer que esa implementación parezca ser un objeto cuyos métodos puedan ser invocados desde una máquina remota.

Un método común es utilizar un adaptador de objeto, el cual actúa como envolvente alrededor de la implementación con el único propósito de darle la apariencia de un objeto.

Para hacer la envoltura tan fácil como sea posible, los objetos se definen únicamente en función de las interfaces que implementan. La implementación de una interfaz puede entonces ser registrada en un adaptador, el cual posteriormente hace que la interfaz esté disponible para invocaciones (remotas).

El adaptador se encargará de que las solicitudes de invocación sean atendidas y, por tanto, de proporcionar una imagen de objetos remotos a sus clientes.

Objetos persistentes y transitorios

Además de la distinción entre objetos a nivel de lenguaje y objetos en tiempo de ejecución, también existe una distinción entre objetos persistentes y objetos transitorios.

Un objeto persistente es uno que continúa existiendo aun cuando ya no esté contenido en el espacio de dirección de cualquier proceso de servidor.

En otros términos, un objeto persistente no depende de su servidor. En la práctica, esto significa que el servidor que actualmente está manejando el objeto persistente puede guardar su estado en un almacenamiento secundario y luego salir.

un objeto transitorio es un objeto que existe sólo en tanto el servidor que lo está alojando exista.
En cuanto el servidor deja de funcionar, el objeto deja de existir.