Tema 8

Comunicación

Características de los mecanismos de comunicación

Los mecanismos de comunicación permiten la comunicación entre procesos que se ejecutan en ordenadores distintos y son ofrecidos por los middlewares de la comunicación

Utilización

Con primitivas básicas de comunicación tenemos operaciones de envio y recepción, como por ejemplo los sockets. Mediante construcciones del lenguaje de programación conseguimos mayor nivel de abstracción y que el envío y recepción de mensajes sea transparente.

Estructura y contenido de los mensajes

Mensajes no estructurados

Solo contenido que estará en formato libre.

Mensajes estructurados en cabecera + contenido

La cabecera es un conjunto de campos extensible, el contenido formato libre.

Estructura transparente al programador

Determinada por el middleware de comunicaciones.

Direccionamiento

Direccionamiento directo

El ordenador emisor envía los mensajes directamente al ordenador receptor.

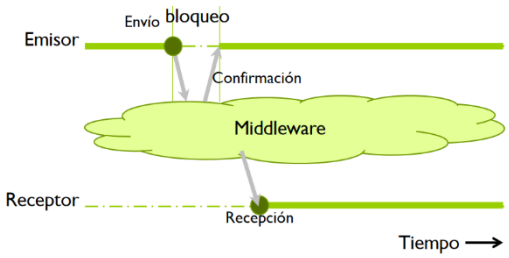
Direccionamiento indirecto

El ordenador emisor envia los mensajes a un intermediario (broker) y este los hace llegar al receptor.

Sincronización

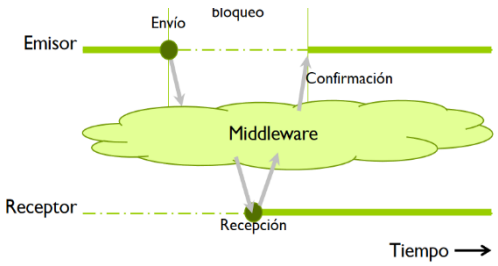
Comunicación asíncrona

El middleware responde al emisor con la confirmación tras almacenarlo en sus buffers



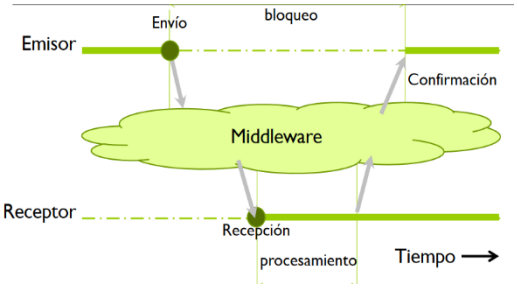
Comunicación síncronica (entrega)

El middleware responde cuando el receptor ha confirmado la entrega correcta del mensaje



Comunicación sincrónica (respuesta)

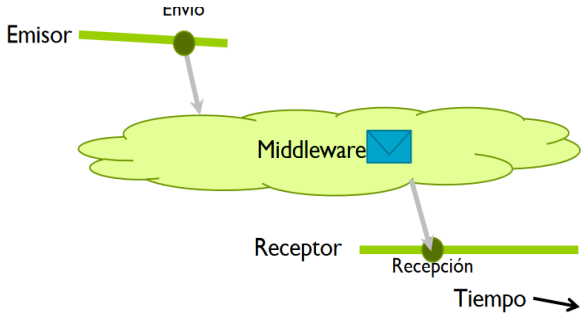
El middleware responde al emisor tras recibir aviso del receptor de haber procesado el mensaje



Persistencia

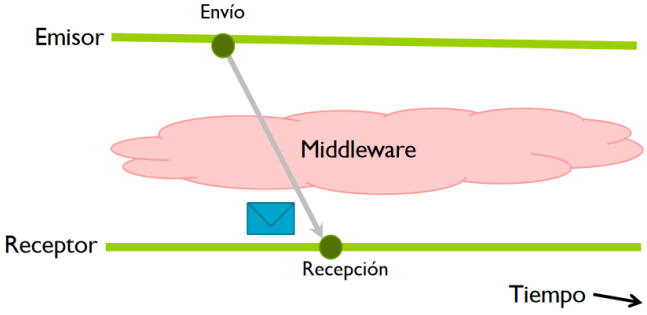
Comunicación persistente

El middleware puede guardar los mensajes pendientes de entrega. El receptor no tiene por qué estar en ejecución cuando se envía el mensaje. El emisor puede finalizar su ejecución antes de que el mensaje se entregue.



Comunicación no persistente

El middleware no es capaz de mantener los mensajes que deben transmitirse. Se requieren el emisor y el receptor activos para que los mensajes lleguen a transmitirse.



Invocación a objeto remoto (ROI)

Un objeto remoto es un objeto que puede ser invocado desde otros espacios de direcciones. Se instancia en un nodo servidor y puede responder a la invocación de clientes locales y remotos.

Como ventajas del modelo objeto remoto tenemos:

- **Aprovechar la expresividad**
- **Transparencia de ubicación**
- **Aplicaciones heredadas**
- **Escalabilidad**

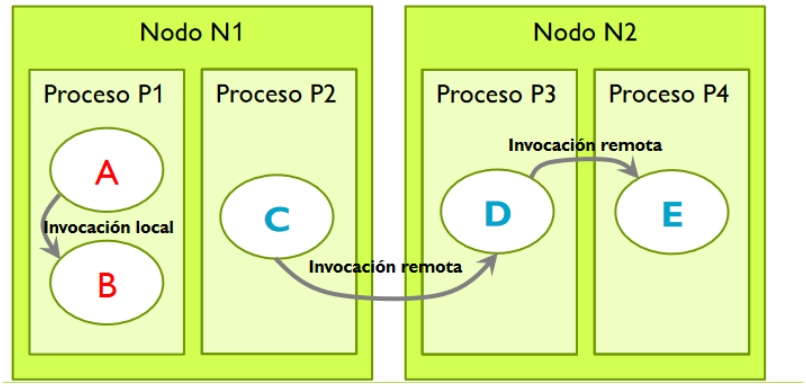
Tipos de invocaciones

Invocación local

Los objetos invocador e invocado residen en el mismo proceso.

Invocación remota

Los objetos invocador e invocado residen en procesos diferentes, dentro del mismo nodo o en nodos distintos.



Antecedentes (Remote Procedure Call)

RPC es el antecedente de ROI, comparte los mismos objetivos. No contempla el concepto de objeto y el ordenador remoto ofrece un catálogo de procedimientos.

Pasos RPC

1- **Invocacion del procedimiento local**
2- **Empaquetado de argumentos de entrada**
3- **Envio del mensaje al servidor y espera de respuesta**
4- **Desempaquetado del mensaje y extracción de argumentos de entrada**

5- Llamada al procedimiento
6- Ejecución del procedimiento
7- Retorno de control al stub servidor
8- Empaquetado de argumentos de salida y resultado en un mensaje
9- Envío del mensaje de respuesta
10- Desempaquetado del mensaje y extracción de argumentos de salida y resultado
11- Retorno de control al código que invocó el procedimiento local

Elementos que intervienen en una ROI

Proxy

Representa al objeto remoto, ofrece la misma interfaz que el objeto remoto. Contiene una referencia a este y a su interfaz´.

Esqueleto

Recibe las peticiones de los clientes. Realiza las verdaderas llamadas a los métodos del objeto remoto

Object request broker (ORB)

Componente principal de un middleware orientado a objetos

1- Identificar y localizar los objetos
2- Realizar las invocaciones remotas de los proxies
3- Gestionar el ciclo de vida de los objetos

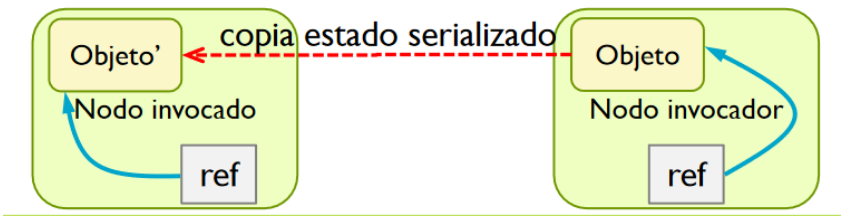
Pasos de una ROI

El cliente invoca el método en el proxy y este se encarga de empaquetar los argumentos. Llama al ORB y gestiona la invocación, el esqueleto invoca al método real, al finalizar librea al esqueleto. Este, llama al ORB empaquetando los argumentos y el proxy hace que le lleguen los resultados al Cliente.

Paso de objetos como argumentos

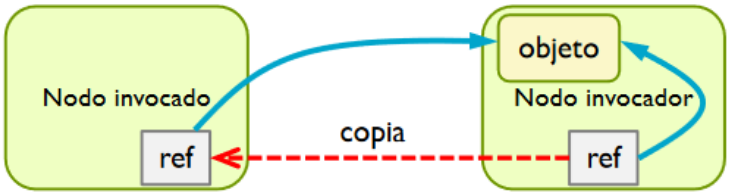
Paso por valor

El estado del objeto origen se empaqueta, mediante la serialización. El objeto serializado se transmite al nodo destino, donde se crea un nuevo objeto copia del original.



Paso por referencia

Para pasar un objeto por referencia basta con copiar la referencia del nodo invocador al nodo invocado.



Creación de objetos ROI

Por iniciativa de cliente

El cliente solicita a una factoria crear el objeto. La factoria crea el objeto solicitado y registra el ORB

La factoria es un objeto servidor que crea objetos de un tipo determinado

Por iniciativa de servidor

Un proceso servidor crea un objeto y lo registra en el ORB, obteniendo una referencia al objeto. El proceso servidor registra la referencia en el servidor de nombres.

Java RMI (Remote Method Invocation)

Middleware de comunicación orientado a objetos que proporciona una solución para un lenguaje OO especifico.

RMI en Java

Incorporación

El componente RMI se incorpora de forma automática a un proceso Java cuando se utiliza su API. Escucha peticiones que llegan en un puerto TCP.

Objeto invocable

Un objeto es invocable de forma remota si implementa una interfaz que extiende la interfaz Remote. Para cada uno de estos objetos remotos, RMI crea dinámicamente un objeto (no accesible por el usuario) llamado esqueleto.

Invocar a un objeto

Para invocar a un objeto remoto desde otro proceso se utiliza un objeto Proxy. Su interfaz es idéntica a la del objeto remoto.

Servidor de nombres de Java RMI

El servidor de nombres almacena, para cada objeto el nombre simbólico + referencia. Puede residir en cualquier nodo y es accedido desde el cliente o servidor utilizando Registry.

Bind

Registrar un objeto con un nombre, si ya existe da una excepción.

Rebind

Igual que Bind pero si ya existía el objeto solo cambia su nombre.

Unbind

Elimina el registro asociado al nombre

Desarrollo de una aplicación Java RMI

Reglas para programar objetos remotos en Java

Interfaz objeto remoto

```
Public interface Hola extends Remote{
  String saluda() throws RemoteException;
}
```

Clase de objetos remotos

```
class ImpleHola extends UnicastRemoteObject implements Hola {
  ImpleHola() throws RemoteException {..} // constructor
  public String saluda() throws RemoteException {
    return "Hola a todos";
  }
}
```

Características

Paso de objetos

- **Implementan la interfaz remote:** se pasa por referencia. El objeto es compartido por las referencias previas y la nueva
- **No implementan la interfaz remote:** Se serializa y se pasa por valor, se crea un objeto copia en la maquina virtual destino totalmente independiente al original.

Servicios Web

Son Sistemas software diseñados para proporcionar interacciones ordenador-ordenador utilizando la red.

Variantes

Servicios web basados en SOAP y WSDL

Son generalmente conocidos como servicios web, SOAP es una especificación XML que se intercambia y WSDL es una especificación XML que describe la funcionalidad ofrecida.

Servicios web RESTful

Alternativa simple y flexible. Se destaca JSON por ser el formato mas habitual para el intercambio de información ya que no se requiere ningún lenguaje de descripción de funcionalidad.

Son muy importantes para el desarrollo de aplicaciones web, Están disponibles en la inmensa mayoría de lenguajes de programación y frameworks de desarrollo. Además, son sencillos. Se accede a ellos mediante URIs, se usa la arquitectura cliente-servidor.

Características de la comunicación en REST

Es un mecanismo de difícil categorización en función de la manera en la que se analice.
Desde el punto de vista del mecanismo subyacente, sus características serian aquellas asociadas a la comunicación basada en el protocolo HTTP y en el uso de sockets.
Desde el punto de vista del uso, es un mecanismo de petición-respuesta.

Middlewares orientados a mensajeria (MOMs)

Son middlewares que ofrecen comunicación basada en mensajes. Los emisores envían no directamente al receptor, sino a un elemento intermedio denominado cola. Una vez depositado el mensaje en la cola, el emisor prosigue con su ejecución sin esperar a que el receptor recoja el mensaje. El receptor recoge el mensaje en cualquier momento. Comunicación asíncrona.

Basados en bróker

La mayoría de estos sistemas están basados en un bróker de comunicaciones, es decir, un proceso servidor que gestiona totalmente las colas, que crea y borra los elementos de esta atendiendo a los envíos y las peticiones de los emisores y receptores.

Java Message Service 2.0 (JMS)

Es una API Java que permite a las aplicaciones enviar y recibir mensajes. Con direccionamiento a través de proveedor, sincronización asíncrona y persistente.

JMS componentes

Proveedores JMS

Sistema de mensajería que implementa las interfaces de JMS y proporciona herramientas administrativas y de control.

Clientes JMS

Programa o componente escrito en Java que produce o consume mensajes.

Mensajes

Objetos que mandan información entre clientes

Objetos administrados

Factorias de conexiones y destinos creados mediante las herramientas administrativas del proveedor JMS.

ZeroMQ

JMS Modelo de programación

Interface ConnectionFactory

Vinculan la aplicación con un objeto administrado y Crean conexiones con el proveedor JMS

Interface JMSContext

Mantienen una conexión con el proveedor JMS y Son utilizables por un solo hilo. Procesan envíos y recepciones de forma secuencial.

Interface JMSProducer

Permiten enviar mensajes a colas y temas.

Interface JMSConsumer

Permiten recibir mensajes de colas y temas.

Interface Destination

Vinculan la aplicación con un objeto administrado.

Interface Message

Son los mensajes que se envían y reciben en JMS.