

Computación distribuida

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid



Contenidos

1. Qué es computación distribuida
2. Principales paradigmas
 - a) Paso de mensajes
 - b) Cliente/Servidor y *Peer-to-Peer*
 - c) Procedimientos remotos y métodos remotos
 - d) Servicios de red, *Object Request Broker* y agentes móviles
 - e) Espacio de objetos y aplicaciones colaborativas

Contenidos

1. **Qué es computación distribuida**
2. Principales paradigmas
 - a) Paso de mensajes
 - b) Cliente/Servidor y *Peer-to-Peer*
 - c) Procedimientos remotos y métodos remotos
 - d) Servicios de red, *Object Request Broker* y agentes móviles
 - e) Espacio de objetos y aplicaciones colaborativas

Sistema distribuido

- ▶ “Sistema en el cual componentes de hardware y software, localizadas en computadores en red, se comunican y coordinan sus acciones sólo por paso de mensajes”

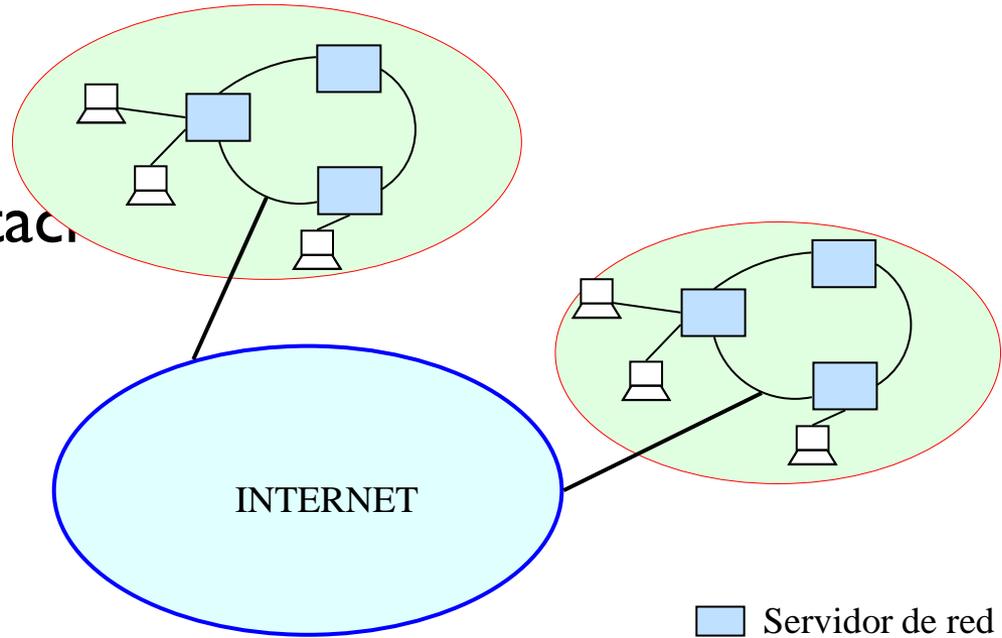
[Coulouris 2002]

- ▶ “Conjunto de computadores independientes que se muestran al usuario como un sistema único coherente”

[Tanenbaum 2001]

Computación distribuida

- ▶ Sistema distribuido
- ▶ Esquemas de computación
 - ▶ Monolítica
 - ▶ Paralela
 - ▶ Distribuida
 - ▶ Cooperativa



- ▶ Computación distribuida:
 - ▶ Servicio de red
 - ▶ Aplicación de red

Computación distribuida

▶ **Ventajas:**

- ▶ Reducción del coste del computador y del acceso a la red.
- ▶ Compartición de recursos.
- ▶ Escalabilidad.
- ▶ Tolerancia a fallos.

▶ **Inconvenientes:**

- ▶ Múltiples puntos de fallo.
- ▶ Seguridad.

Computación distribuida

- ▶ **Conectividad** (usuarios y recursos)
- ▶ **Desempeño** (tiempo de respuesta, productividad)
- ▶ **Robustez** (disponibilidad y consistencia)
- ▶ **Seguridad** (autenticación, privacidad y control de acceso)
- ▶ **Transparencia** (ubicación, acceso, fallas, partición, replicación, migración, etc.)
- ▶ **Escalabilidad** (tamaño, distancia y gestión)
- ▶ **Apertura (*Openness*)** (interfaces, interoperabilidad y portabilidad)

Contenidos

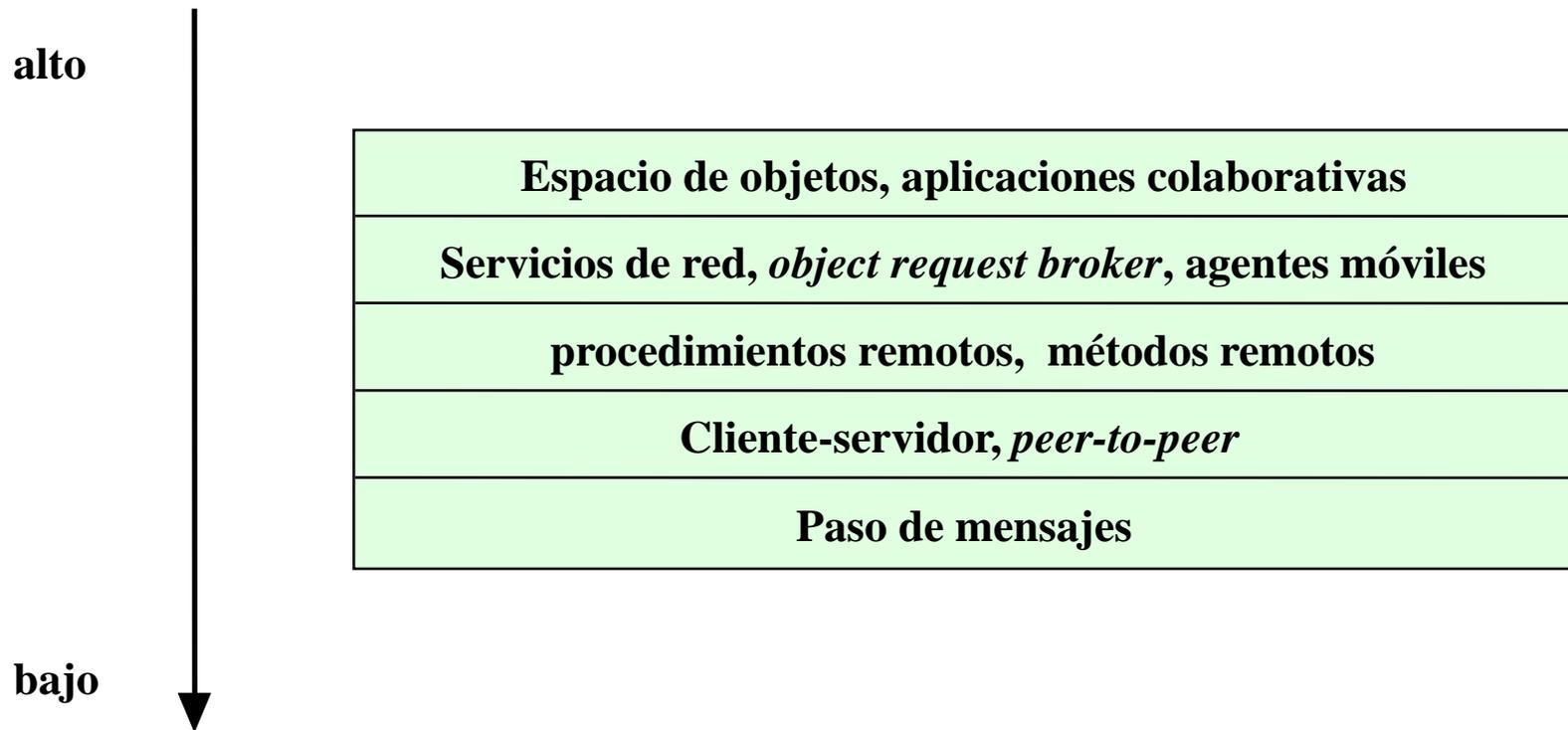
1. Qué es computación distribuida
2. **Principales paradigmas**
 - a) Paso de mensajes
 - b) **Cliente/Servidor y *Peer-to-Peer***
 - c) **Procedimientos remotos y métodos remotos**
 - d) **Servicios de red, *Object Request Broker* y agentes móviles**
 - e) **Espacio de objetos y aplicaciones colaborativas**

Paradigmas de computación distribuida

- ▶ **Abstracción:** encapsulación o ocultamiento de detalles.
- ▶ **Paradigma:** “*un patrón, ejemplo o modelo*”.
 - ▶ Estrategia: identificar los patrones o modelos básicos y clasificar los detalles de acuerdo con estos modelos.
- ▶ **Características de las aplicaciones distribuidas:**
 - ▶ *Comunicación entre procesos:* una aplicación distribuida requiere la participación de dos o más procesos.
 - ▶ *Sincronización de eventos:* deben de existir mecanismos de sincronización para el correcto envío y recepción de la información.

Paradigmas de computación distribuida

- ▶ Los paradigmas se representan clasificados de acuerdo con su nivel de abstracción.

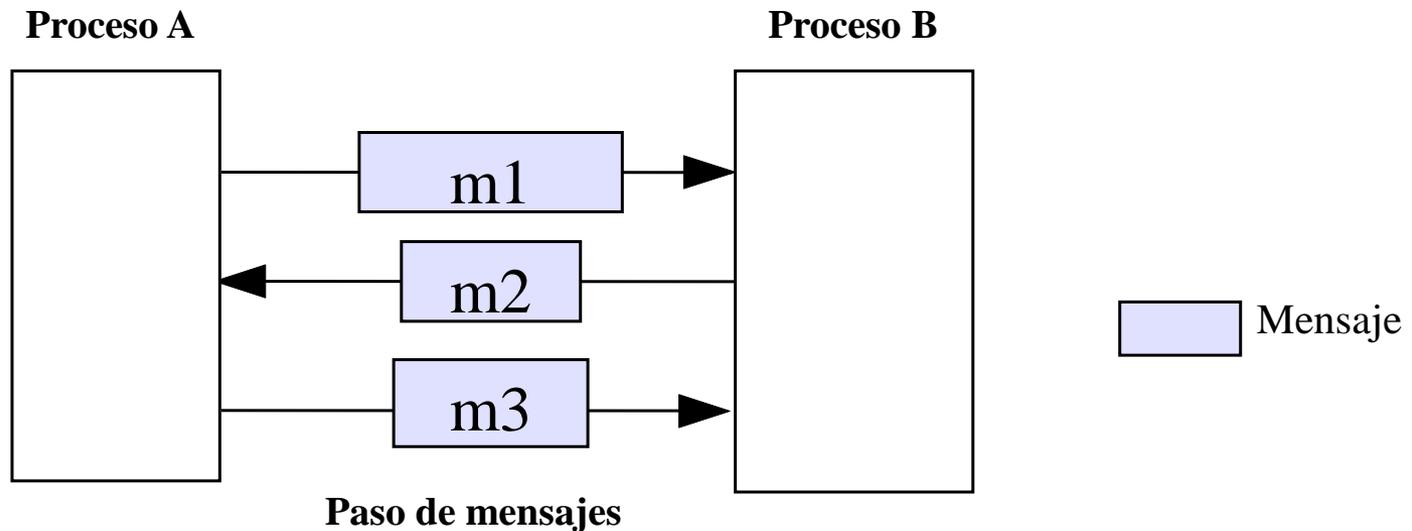


Paradigma de paso de mensajes



Paradigma de paso de mensajes

- ▶ Paradigma fundamental para aplicaciones distribuidas
 - ▶ Un proceso envía un mensaje de solicitud
 - ▶ El mensaje llega al receptor, el cual procesa la solicitud y devuelve un mensaje en respuesta
 - ▶ Esta respuesta puede originar posteriores solicitudes por parte del emisor



Paradigma de paso de mensajes

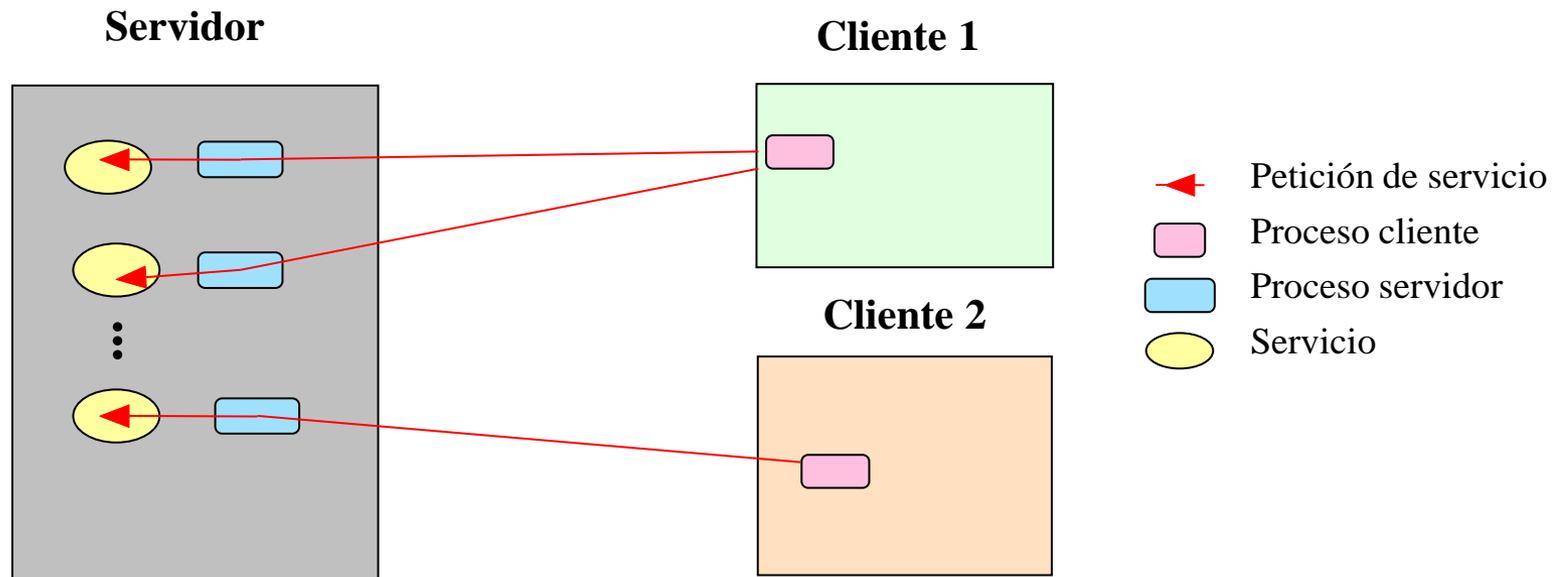
- ▶ Las operaciones básicas para soportar el paradigma de paso de mensajes son *enviar* y *recibir*
 - ▶ Protocolos más comunes: IP y UDP
- ▶ Para las comunicaciones orientadas a conexión también se necesitan las operaciones *conectar* y *desconectar*
 - ▶ Protocolo más común: TCP
- ▶ Operaciones de Entrada/Salida que encapsulan el detalle de la comunicación a nivel del sistema operativo
 - ▶ Ejemplo: el API de *sockets*

Paradigmas cliente/servidor y P2P



Paradigma cliente-servidor

- ▶ Asigna roles diferentes a dos procesos que colaboran:
 - ▶ Servidor: es el proveedor del servicio.
Espera de forma pasiva la llegada de peticiones.
 - ▶ Cliente: invoca peticiones al servidor y aguarda su respuesta.

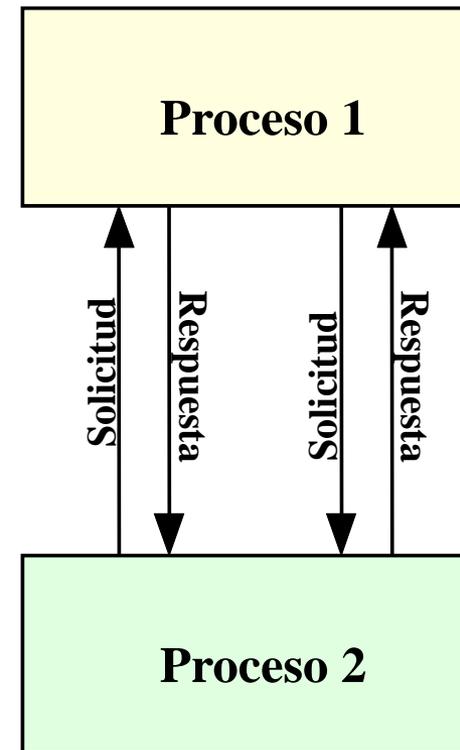


Paradigma cliente-servidor

- ▶ Proporciona una abstracción eficiente para facilitar los servicios de red.
- ▶ La **asignación de roles asimétricos** simplifica la sincronización.
- ▶ Paradigma **adecuado para servicios centralizados**.
 - ▶ **Ejemplos:** servicios de internet como **HTTP, FTP, DNS, finger, etc.**
- ▶ Implementación mediante *sockets*, llamada a procedimientos remotos (RPC) o invocación de métodos remotos (RMI).

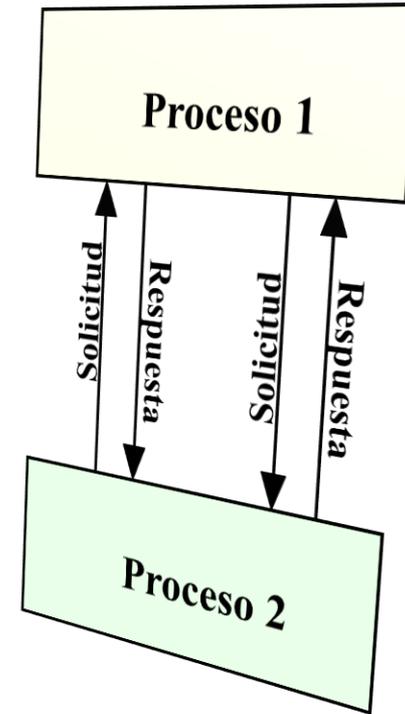
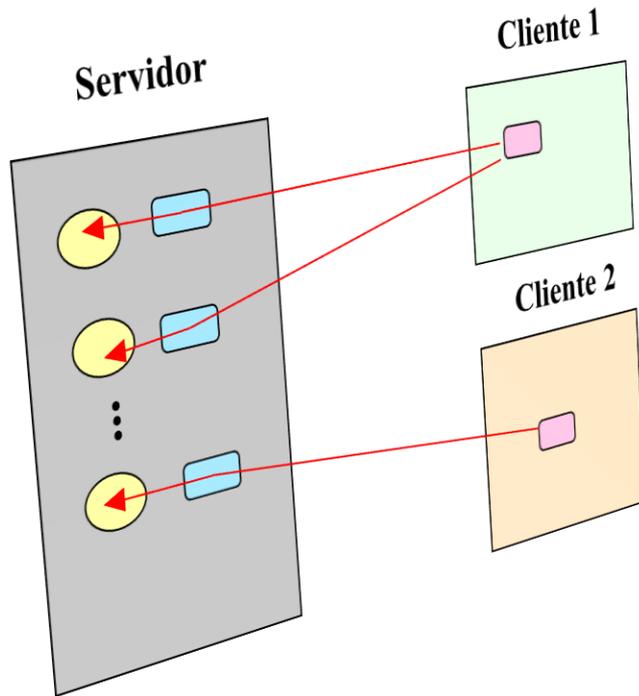
Paradigma *peer-to-peer*

- ▶ **Asignación de roles simétrica:**
 - ▶ Los procesos participantes tienen el mismo papel
 - ▶ un mismo proceso puede actuar tanto como cliente como servidor
- ▶ Los recursos computacionales y los servicios son intercambiados entre los computadores.
 - ▶ **Ejemplo:** servicios de intercambio de ficheros como **Gnutella**



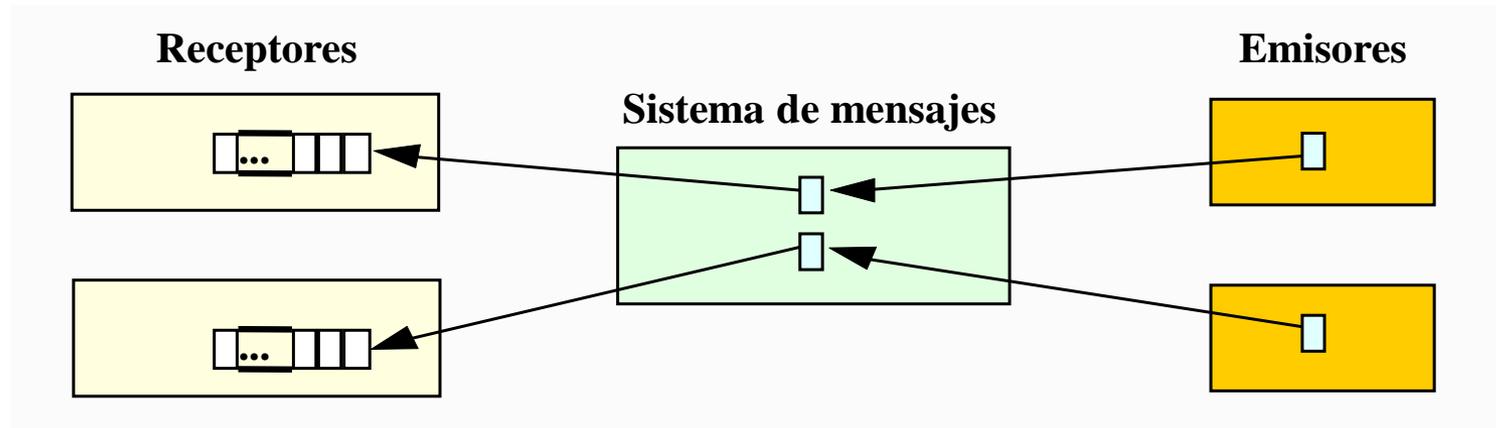
Paradigma *híbridos* (c/s + p2p)

- ▶ Modelos híbridos cliente-servidor y *peer-to-peer*
 - ▶ **Ejemplo:** servicio de intercambio de ficheros **Napster**



Paradigma del sistema de mensajes

- ▶ También denominado *middleware orientado a mensajes* (MOM)
- ▶ El sistema de mensajes actúa de intermediario entre los procesos que se comunican
- ▶ Proceso:
 - ▶ Emisión al sistema de mensajes
 - ▶ Almacenamiento en la cola asociada al receptor
 - ▶ Envío al proceso receptor



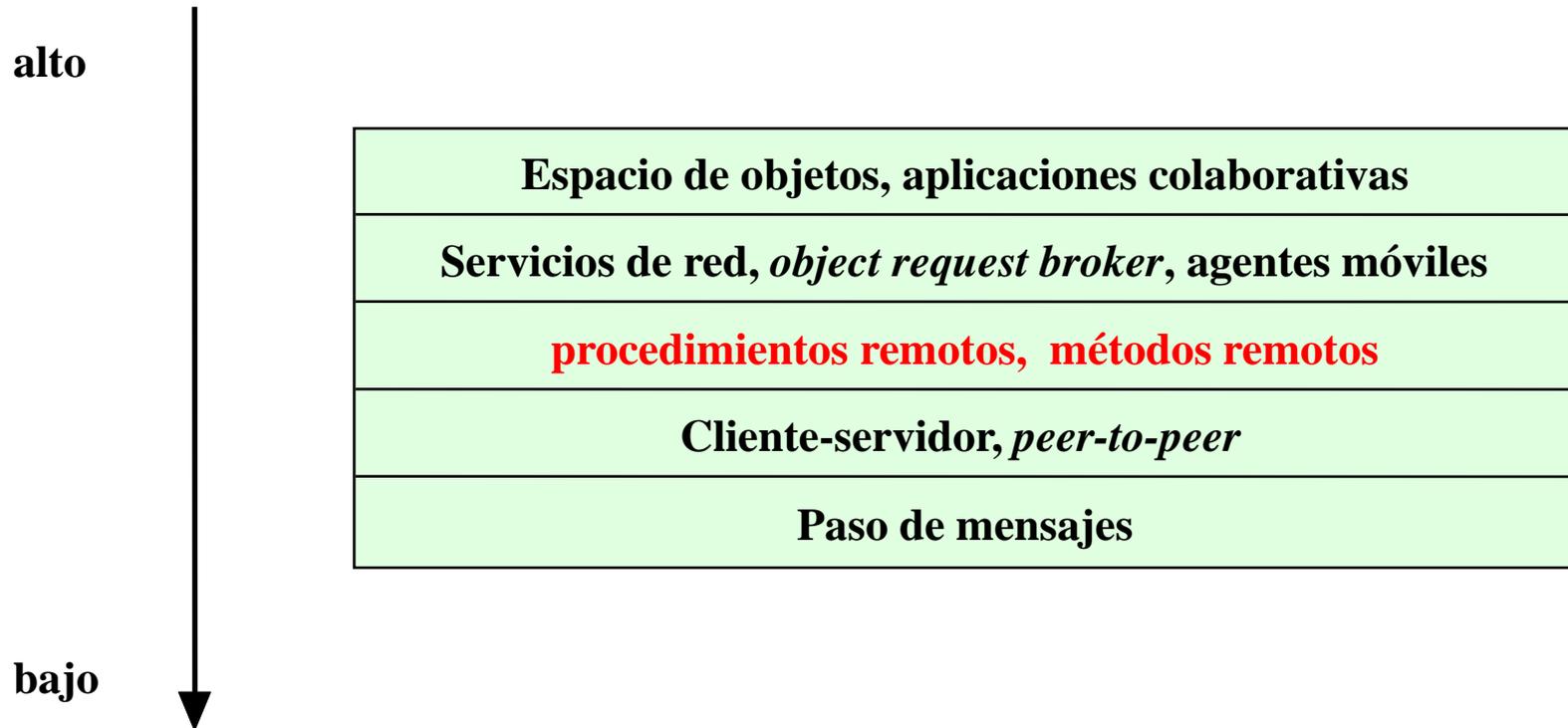
Paradigma del sistema de mensajes

- ▶ Comunicación **asíncrona** y desacoplada.
- ▶ Una vez que el emisor envía el mensaje al sistema de mensajes, queda libre para realizar otra tarea.
- ▶ Existen dos subclases de sistema de mensajes: el **punto a punto** y el **publicación/suscripción**.
- ▶ Sistema de mensajes punto a punto:
 - ▶ El sistema de mensajes proporciona el *middleware* que gestiona cada cola de mensajes
 - ▶ Envío y recepción están desacopladas: uso del *threads* o procesos hijo

Paradigma del sistema de mensajes

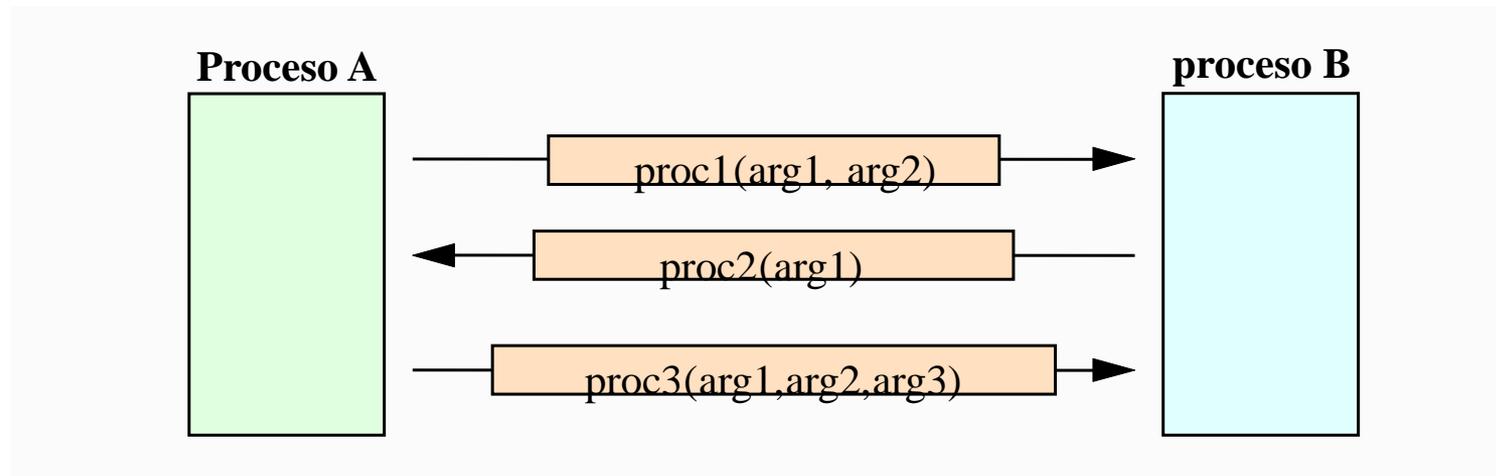
- ▶ Sistema de mensajes publicación/suscripción:
 - ▶ Cada mensaje se asocia con un determinado evento.
 - ▶ Pasos:
 1. Cada participante se suscribe a los mensajes asociados a cada evento (operación *suscribir*).
 2. Cuando el evento ocurre el *middleware* distribuye el mensaje a todos los subscriptores (operación *publicar*).
 - ▶ Los eventos pueden ser iniciados por cualquier participante.
- ▶ Ejemplos de servicio:
 - ▶ *MQ*Series* de IBM
 - ▶ *Microsoft's Message Queue* (MSMQ)
 - ▶ *Java's Message Service* (JMS)

Paradigmas de procedimientos/métodos remotos



Llamadas a procedimientos remotos

- ▶ **Objetivo:** hacer que el software distribuido se programe igual que una aplicación no distribuida.
- ▶ Mediante el modelo RPC la comunicación se realiza conceptualmente igual que la invocación de un procedimiento local.



Llamadas a procedimientos remotos

▶ Pasos:

- ▶ **A** llama al procedimiento remoto de **B**.
- ▶ La llamada dispara una acción de un procedimiento de **B**.
- ▶ Al finalizar el procedimiento, **B** devuelve el valor a **A**.

▶ Simplifica la comunicación entre procesos y la sincronización de eventos.

▶ Ejemplos:

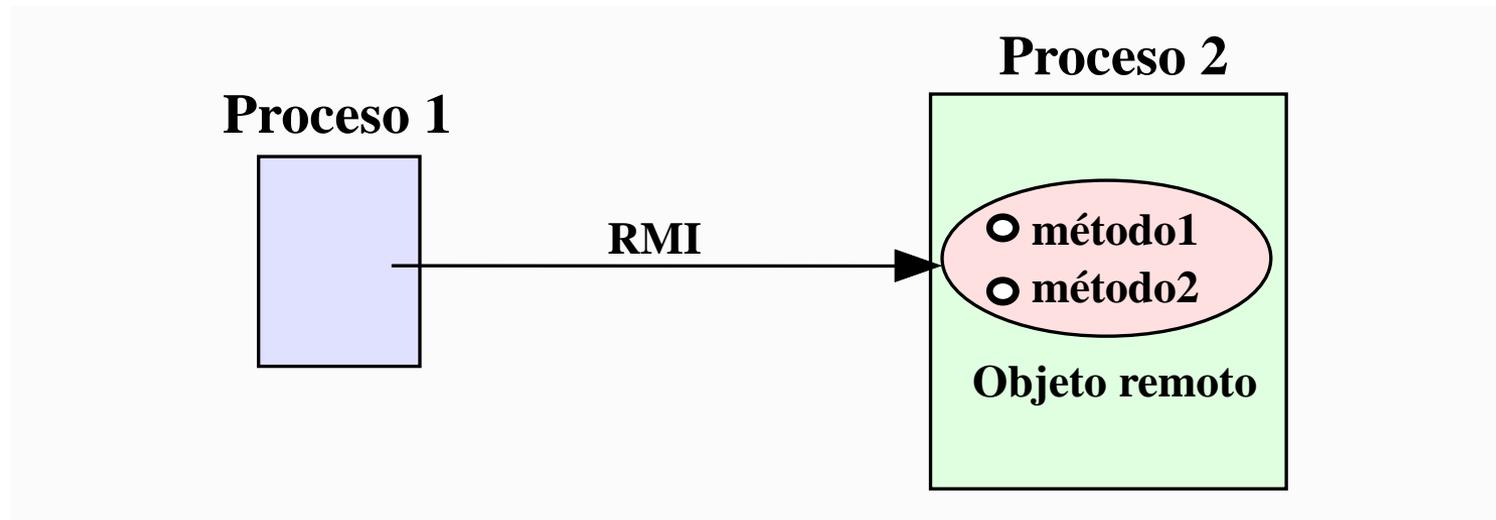
- ▶ *Open Network Computing Remote Procedure Call*, desarrollada a partir del API RPC de *Sun Microsystems* a comienzo de los años 80
- ▶ *Distributed Computing Environment (DCE) RPC* de *Open Group*
- ▶ *Simple objeto Access Protocol (SOAP)*

Llamada a métodos remotos

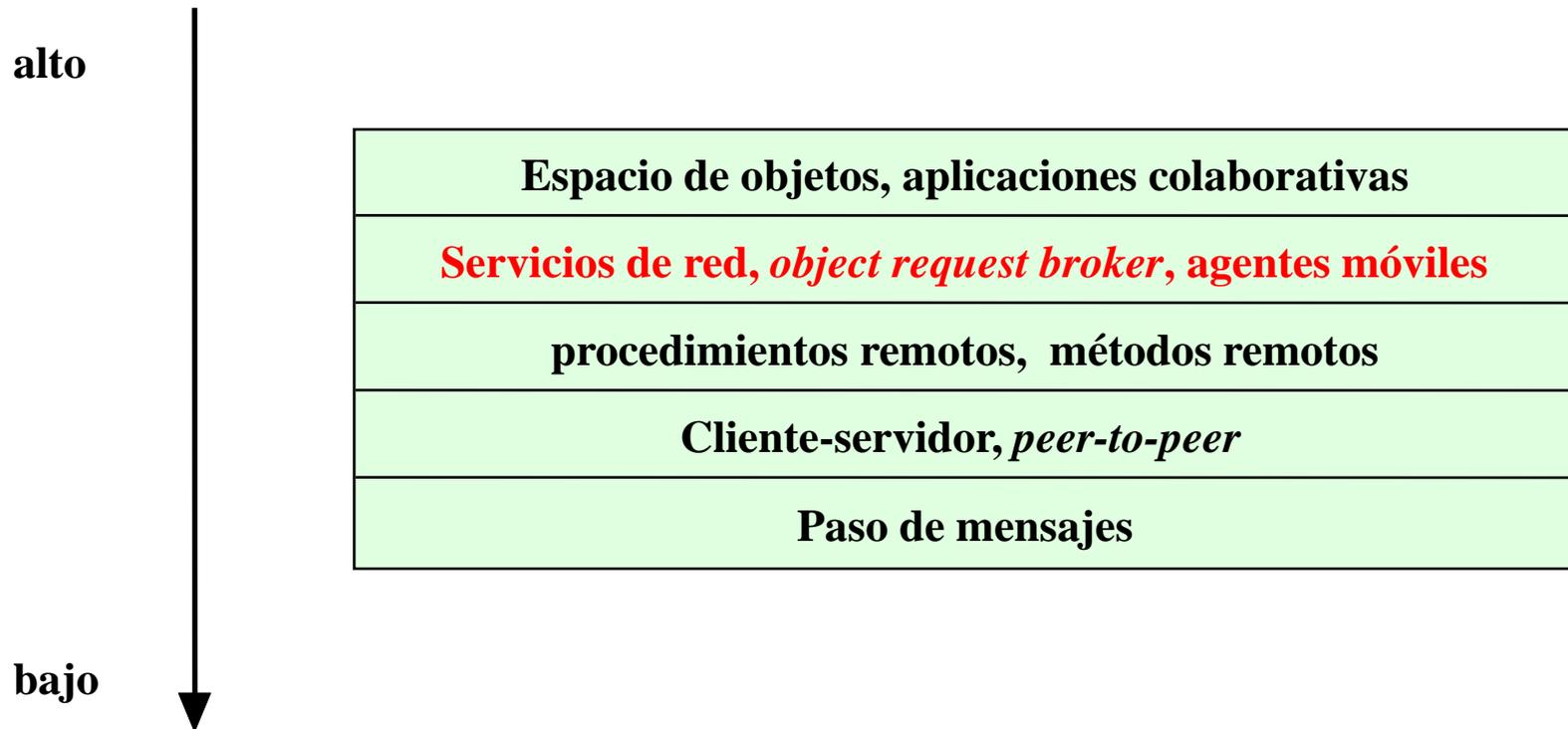
- ▶ Primera aproximación al uso de un modelo orientado a objetos sobre aplicaciones distribuidas
- ▶ **Objetos distribuidos dentro de una red**
 - ▶ Los **objetos proporcionan métodos**, los cuales **dan acceso a los servicios**
- ▶ **Ejemplo:**
 - ▶ *Remote method invocation (RMI)* de Java

Remote method invocation

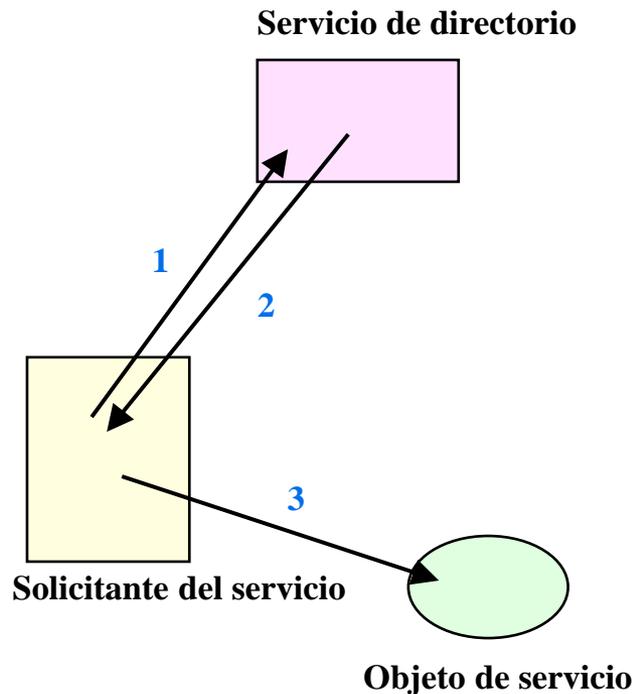
- ▶ Modelo equivalente a las llamadas a procedimientos remotos
- ▶ Proceso invoca un método local de otro proceso
- ▶ Se envían tanto los argumentos del método como el valor devuelto por el mismo



Paradigmas de Servicios de red, ORB y agentes móviles



Paradigma de servicios de red



- ▶ Servicio de directorio: proporcionan la referencia a los servicios disponibles
- ▶ Pasos:
 1. El proceso solicitante contacta con el **servicio de directorio**
 2. El servicio de directorio devuelve la **referencia al servicio solicitado**
 3. Usando la referencia, el proceso solicitante **interactúa** con el **servicio**

Paradigma de servicios de red

- ▶ Extensión del paradigma de invocación de métodos remotos
- ▶ **Transparencia de localización:** nivel de abstracción extra
- ▶ Ejemplos:
 - ▶ Tecnología *Jini* de Java
 - ▶ Protocolo SOAP lo aplica para servicios accesibles en la Web

Paradigma basado en *Object Request Broker*

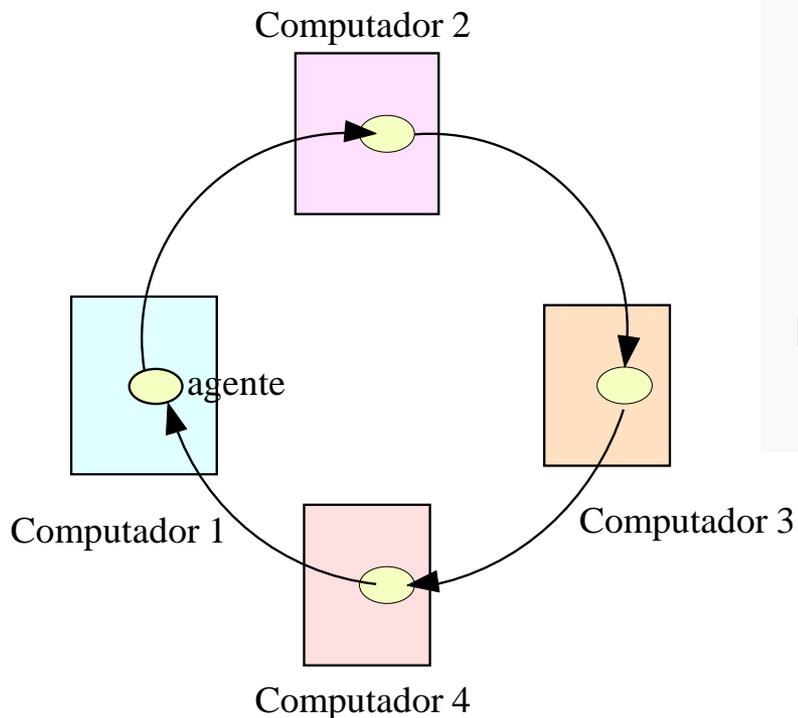
- ▶ El **ORB** funciona como una capa **middleware**.
- ▶ El **ORB** **redirige** las **peticiones al objeto apropiado** que proporciona el servicio solicitado.
- ▶ Extensión a los paradigmas a RMI y servicios de red:
 - ▶ **Instanciación** de clases y objetos



Paradigma basado en *Object Request Broker*

- ▶ El ORB actúa como mediador de objetos heterogéneos
- ▶ Ejemplos:
 - ▶ CORBA (*Common Object Request Broker Architecture*)
 - ▶ Java CORBA
 - ▶ *Visibroker* de Inspire.
 - ▶ *IONA* de Orbix y *TAO* de *Objet Computing, Inc.*
 - ▶ *Microsoft COM, DCOM.*
 - ▶ *Java Beans y Enterprise Java Beans.*

Paradigma de agentes móviles



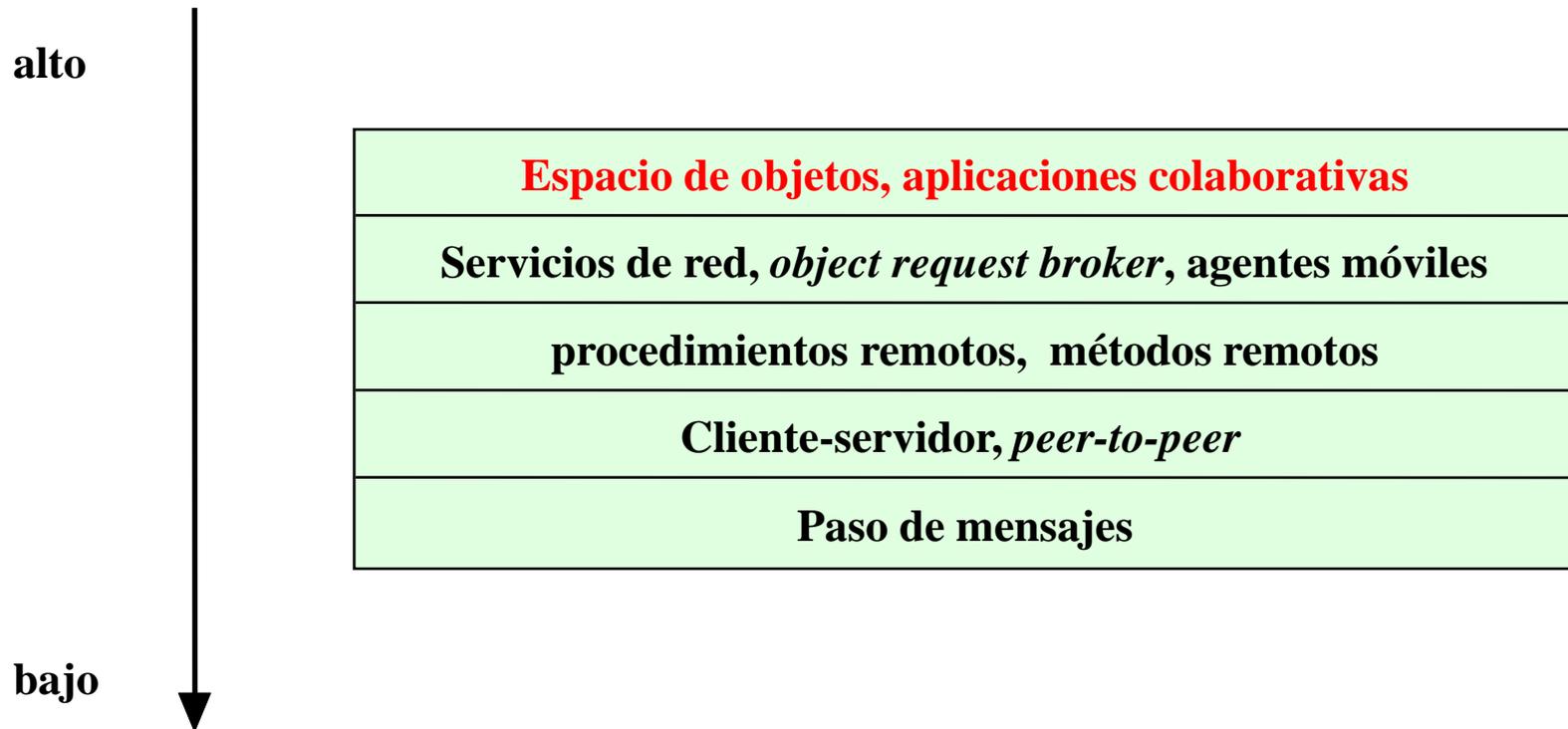
- ▶ **Agente móvil:** programa u objeto transportable.
 - ▶ Un agente se lanza desde un ordenador
 - ▶ Viaja de forma automática de acuerdo con un itinerario
- ▶ Accede a los recursos o servicios de cada sistema que visita

Paradigma de agentes móviles

- ▶ Ejemplos:
 - ▶ Concordia system de Mitsubishi Electric ITA.
 - ▶ Aglet system de IBM.

- ▶ Sistemas experimentales:
 - ▶ D'agent.
 - ▶ Proyecto Tacoma.

Paradigmas de Espacio de objetos y aplicaciones colaborativas

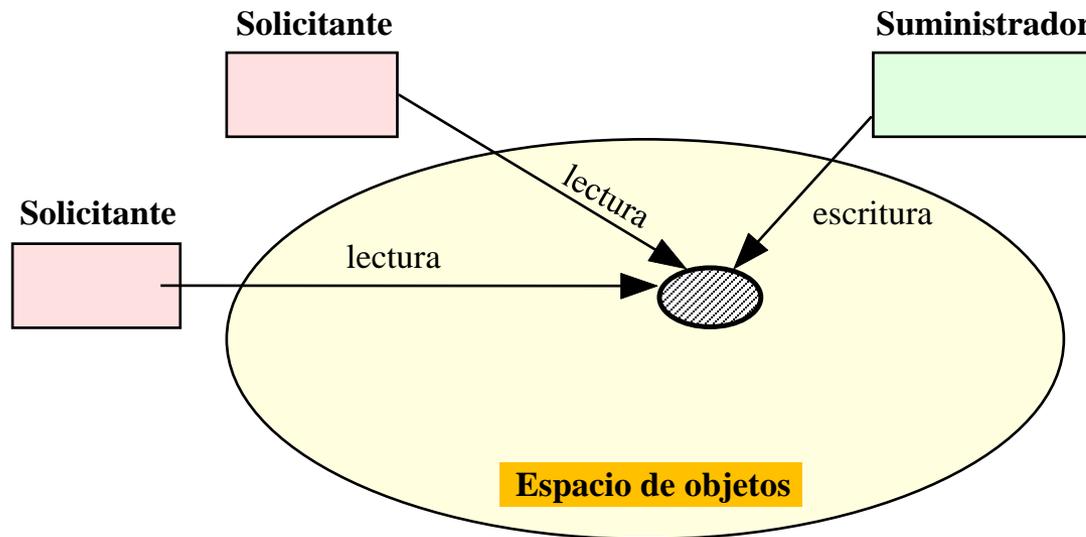


Paradigma de espacio de objetos

- ▶ Denominadas **tecnologías basadas en componentes**.
 - ▶ Objeto empaquetado, con formato y especializado.
- ▶ **Oculto** el nivel de detalle implicado en la **búsqueda** de **recursos y objetos** distribuidos.
- ▶ Elimina la declaración de secuencias de sincronización.

Paradigma de espacio de objetos

- ▶ **Espacio de objetos:** entidades lógicas que contiene un conjunto de objetos comunes a los solicitantes
- ▶ **Suministrador:** provee de objetos al espacio
- ▶ **Solicitantes:** se subscriben al espacio para acceder a los objetos



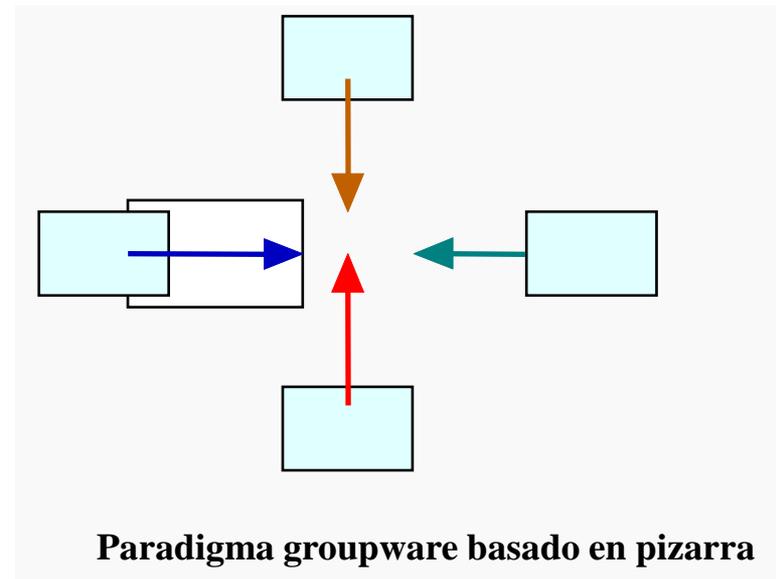
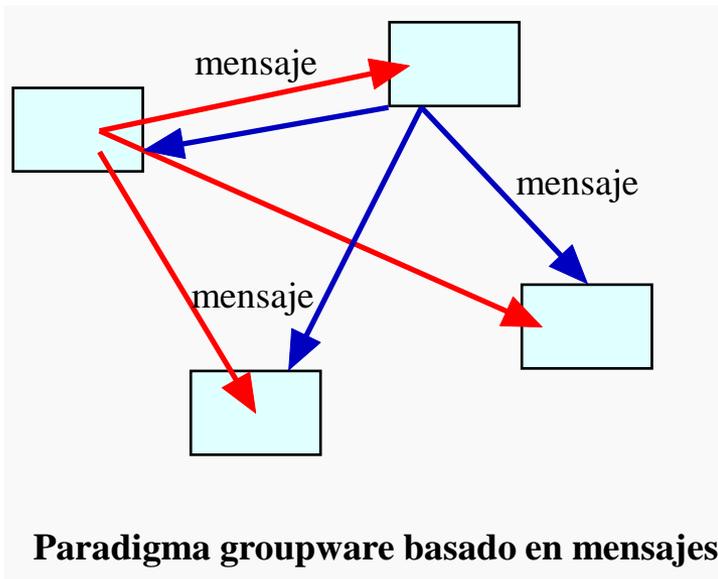
Paradigma de espacio de objetos

- ▶ Espacio virtual o sala de reunión entre los **suministradores** y **solicitantes**.
- ▶ **Un objeto** en el espacio **sólo** puede ser **usado** por un **participante** al mismo **tiempo**.
- ▶ Exclusión mutua queda asegurada.
- ▶ **Solicitantes pueden** actuar como **suministradores**.

- ▶ **Ejemplo:** *JavaSpaces* de SUN.

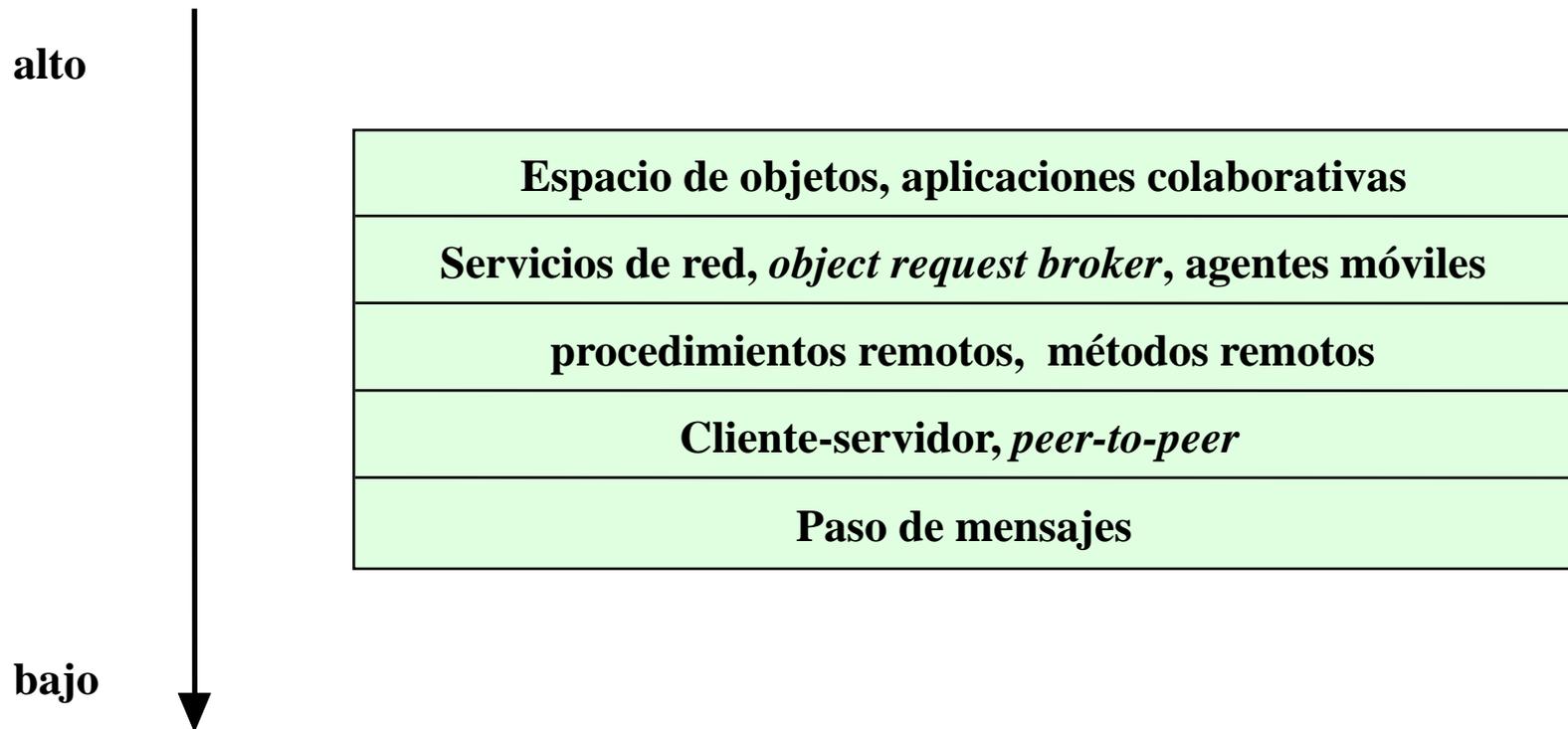
Paradigma de aplicaciones colaborativas

- ▶ **Groupware**: sesión colaborativa en la que participan los procesos.
- ▶ Cada participante puede hacer contribuciones a todo o parte del grupo mediante:
 - a. Multidifusión.
 - b. El empleo de pizarras virtuales.



Paradigmas de computación distribuida

- ▶ Los paradigmas presentados:



Paradigmas de computación distribuida

- ▶ **Pregunta:**

- ▶ ¿Cómo decidir el paradigma más apropiado para una tarea dada?

- ▶ **Respuesta:**

- ▶ Estudio de las ventajas y desventajas de cada uno de ellos.

- ▶ **Factores:**

- ▶ Nivel de abstracción frente a sobrecarga.
 - ▶ Escalabilidad.
 - ▶ Soporte multi-plataforma.
 - ▶ Otras consideraciones: estabilidad de la herramienta, tolerancia a fallos, disponibilidad de herramientas de desarrollo, reutilización del código, etc.



Computación distribuida



Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid