



Tema 4

Invocación Remota

Carlos Montellano



Contents

1. Invocación a Método Remotos - RMI

2. Modelo de Objetos Distribuidos

3. Arquitectura RMI

4. Servicios de Nombres

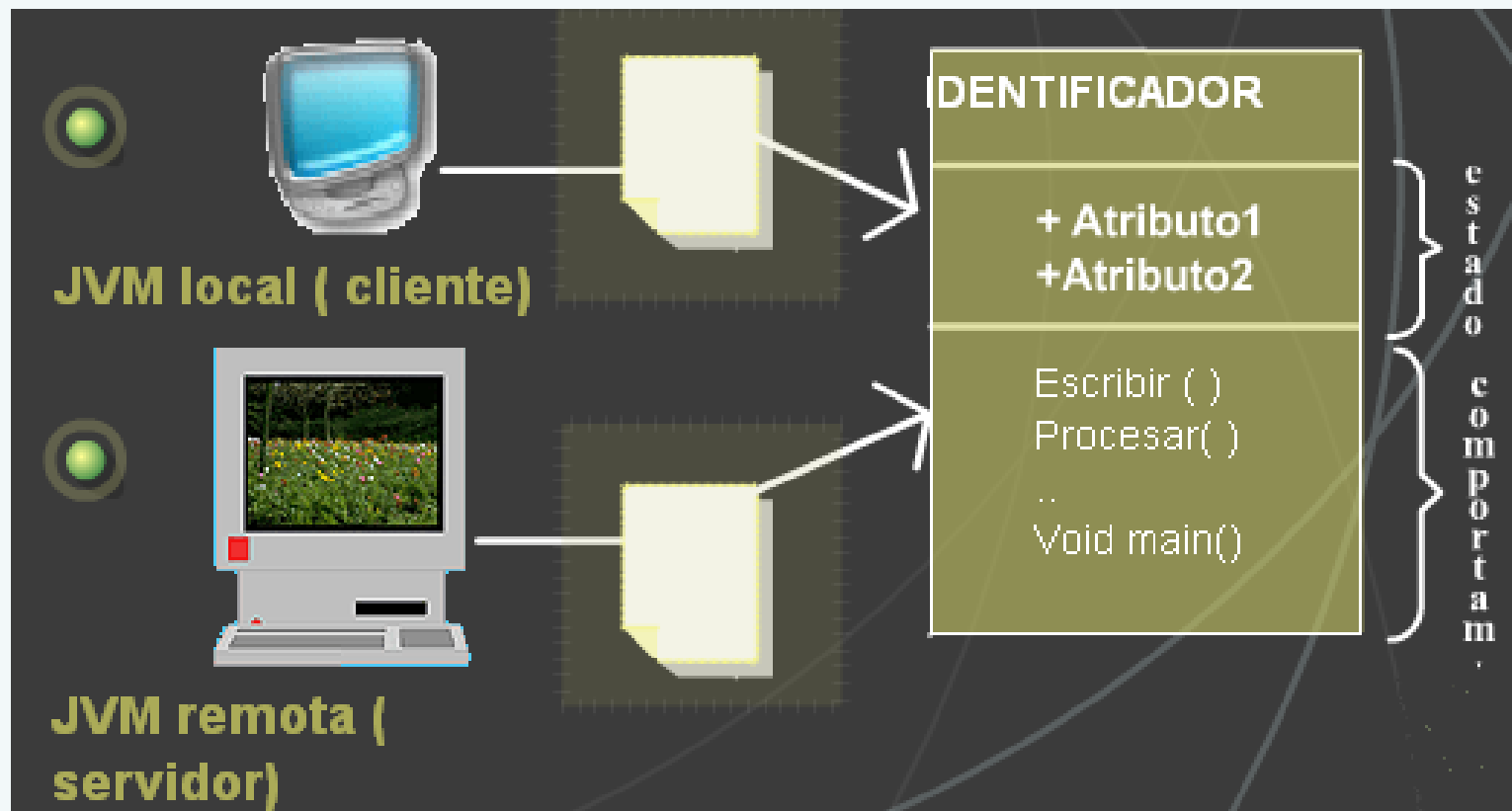
Invocación Remota a Métodos

❖ ¿Qué es RMI?



Modelo de Objetos Distribuido de Java

❖ Modelo de Objetos y Tipo (local y remoto)



A faint, stylized map of the Americas is visible in the background of the slide, showing the outlines of North and South America with some major cities and geographical features.

Modelo de Objetos Distribuido de Java

❖ Características Principales

- Sencillez
- Transparencia
- Paso de Objetos por Valor (como parámetros de los métodos)
- Implementación 100% JAVA
- Independencia del protocolo de comunicación

Modelo de Objetos Distribuido de Java

❖ Los Objetos Distribuidos en Java



Modelo de Objetos Distribuido de Java

❖ Implementación de un Objeto Remoto

- Construir una Clase
- Objetos Remotos
 - Transitorios
 - Permanentes

Modelo de Objetos Distribuido de Java

❖ Modelo de Ejecución

- Crear Objeto del Servidor
- Registrar Servicio
- Esperar Peticiones
 - Recibir petición
 - Averiguar método apropiado (obj.metodoApro(parámetros))
 - Construir mensaje de respuesta
 - Copiar el mensaje resultado del método
 - Enviar mensaje de respuesta
- Forver

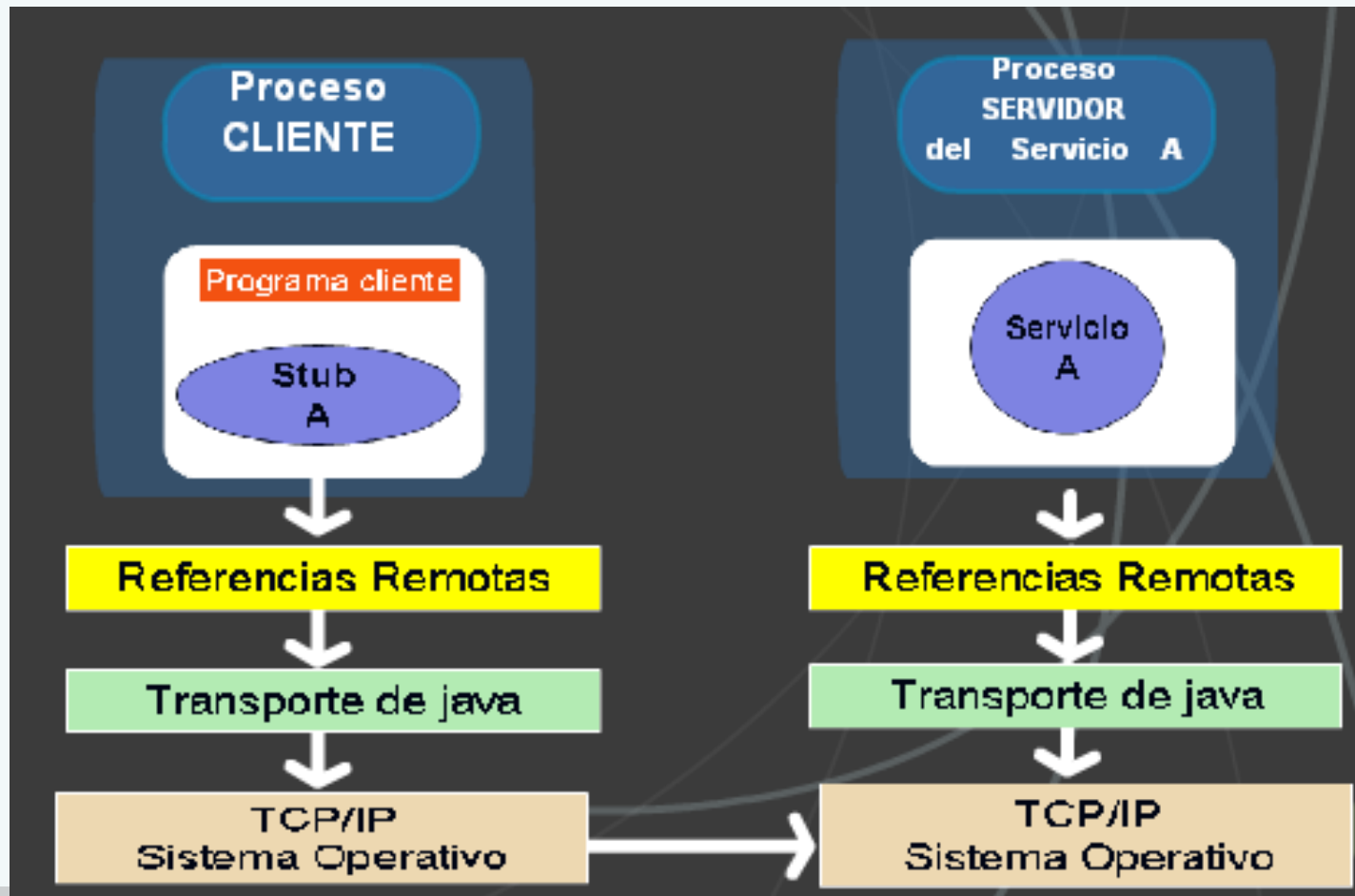
Arquitectura RMI

CAPA STUB

CAPA REFERENCIAS REMOTAS

CAPA DE TRANSPORTE

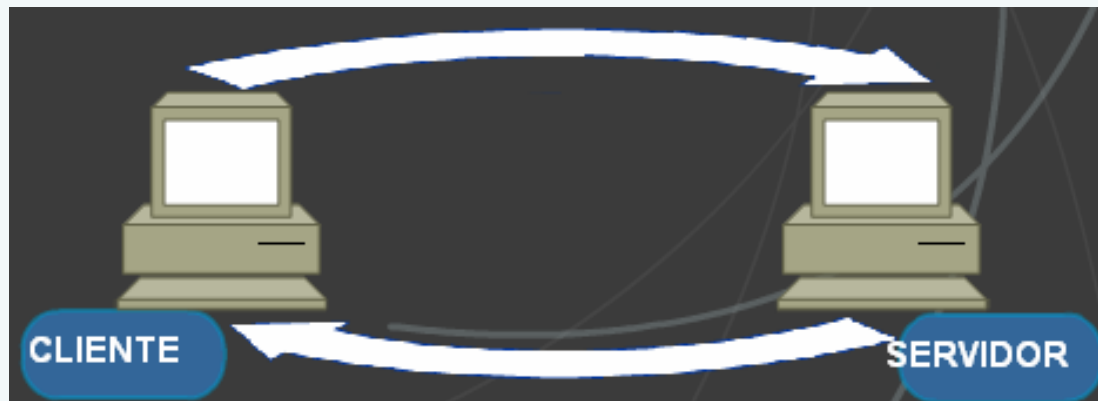
Arquitectura RMI



Arquitectura RMI

❖ La Capa de Stub

- Establecer la comunicación con el Servidor
- Controla dicha comunicación
- Realiza operaciones de serialización y deserialización de parámetros y del resultado.



Arquitectura RMI

❖ Capa de Referencias Remotas

- Operaciones relacionadas con el ciclo de vida de un objeto remoto
- La creación y la destrucción son operaciones que afectan al servidor
- Servidor crea objetos transitorios y permanentes



Arquitectura RMI

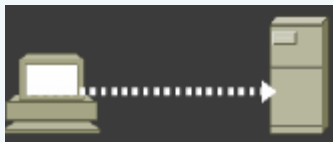


❖ Capa de Referencias Remota

- La semántica de invocación de los objetos a los cliente
- El cliente utiliza dos tipos de semánticas de invocación a métodos remotos estos son:

CLIENTE/SERVIDOR

Comunicación sincrónica clásica



EN GRUPO

Permite que el Stub del Cliente realice peticiones a más de un objeto servidor simultáneamente

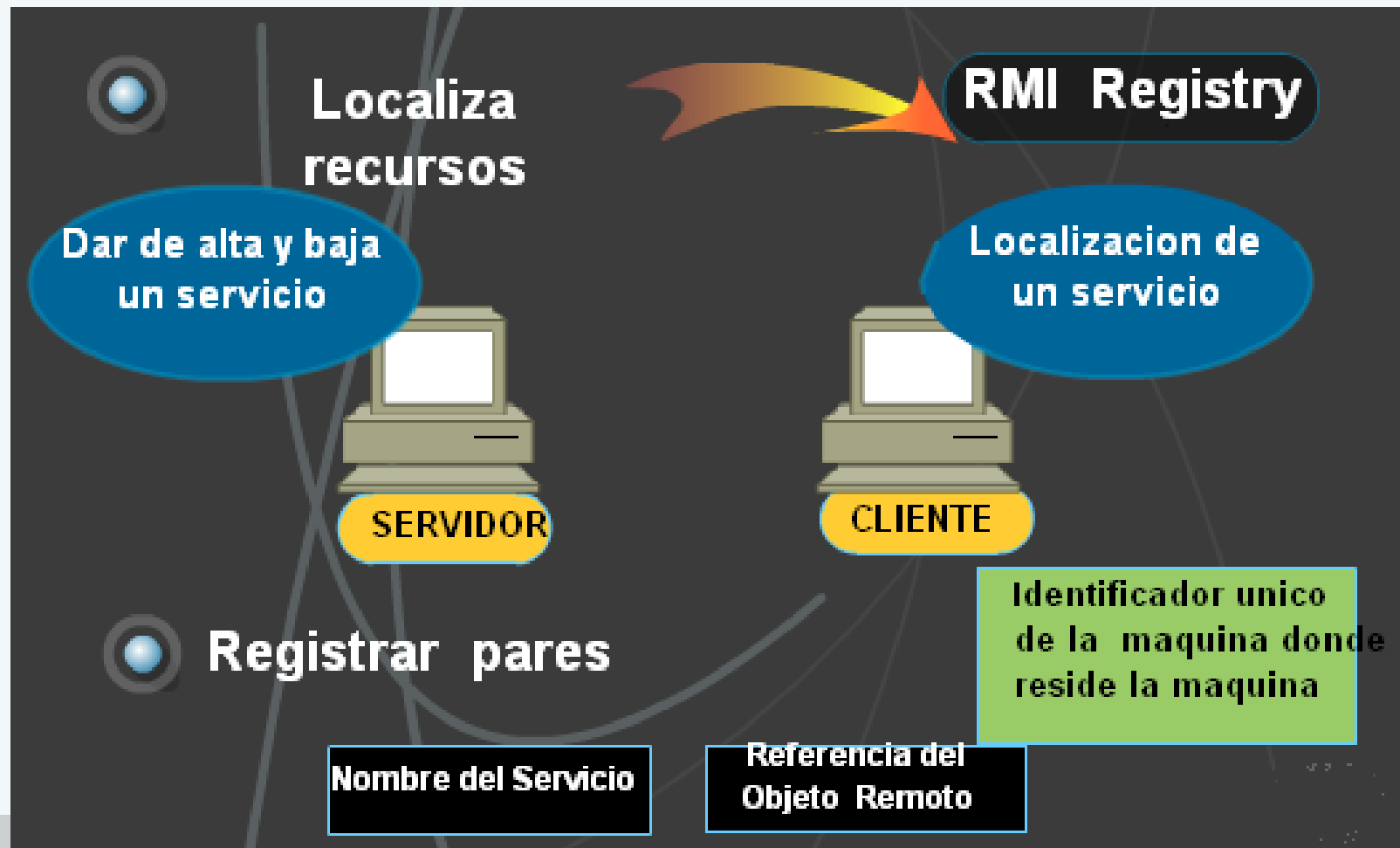


Arquitectura RMI

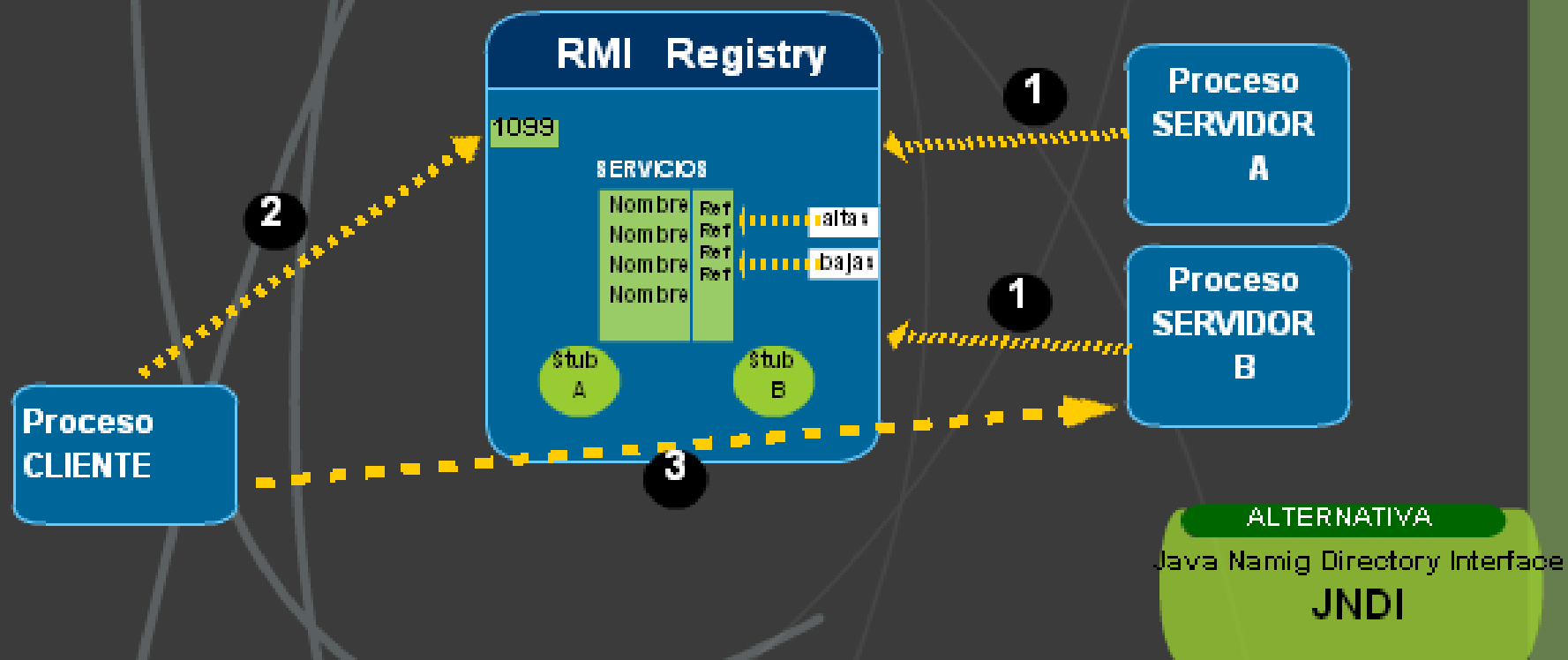
- ❖ **Capa de Transporte - JRMP (*Java Remote Method Protocol*)**
 - Creación de Canales de comunicación
 - Monitorización para comprobar que las conexiones siguen vivas y funcionando correctamente
 - Detectar la llegada de nuevos mensajes ya sean de petición o respuesta
 - Mantener en cada maquina virtual una tabla de los objetos remotos que residen en ella.



Servicio de Nombres



Servicio de Nombres





Ventajas y Desventajas

❖ Ventajas

- Más fácil que RPC y CORBA
- Se puede implementar una aplicación distribuida rápidamente
- Utiliza la misma semántica que el API de Java, por ende ideal para programadores con experiencia en Java
- Ideal para aplicaciones menores, 100% Java

❖ Desventajas

- Existe algunas deficiencias en la funcionalidad
- No soporta aplicaciones multi-lenguaje
- No es apropiado para una aplicación de envergadura empresarial.



RMI vs RPC

❖ RMI

- La lógica del negocio es escrita como objetos cuyos métodos son invocados de forma remota
- Orientado a Objetos y multiplataforma
- Solo puede conectar aplicaciones en Java
- Evolución hacia IIOP (CORBA), multiplataforma y multi-lenguaje

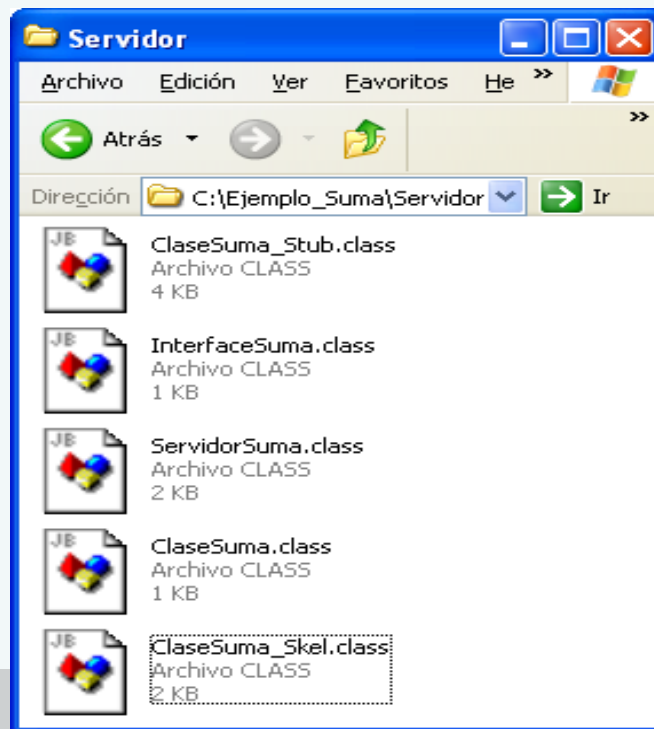
❖ RPC

- Lógica de negocio escrita con procedimientos que son invocados de forma remota
- Obliga a utilizar el mismo lenguaje en ambos lados
- Dependiente de la plataforma
- Poco escalable
- Diseño funcional de servicios, no orientado a objetos.

Ejemplo

- ❖ `javac InterfazSuma.java`
- ❖ `javac ClaseSuma.java`
`rmic ClaseSuma`
- ❖ `start rmiregistry`

`java ServidorSuma`



`java ClienteSuma`

