# SISTEMAS DISTRIBUIDOS

## PROGRAMA
## DE INGENIERIA DE SISTEMAS

ING. DANIEL EDUARDO PAZ PERAFÁN

# CORBA

## Common Object Request Broker Architecture

### Concepto ¿Qué es?

- Arquitectura diseñada para alcanzar la máxima interoperabilidad.
- Conjunto de protocolos que permiten que los programas de aplicación interoperen, con independencia de sus lenguajes de programación, plataformas hardware y software, redes sobre las cuales se comunican y sistemas operativos donde se ejecutan.
  - Distintos sistemas operativos (Unix, Windows, MacOS, OS/2)
  - Distintos protocolos de comunicación (TCP/IP, IPX, …)
  - Distintos lenguajes de programación (Java, C, C++, …)

# CORBA

## ¿Para qué sirve?

Permite la invocación de métodos de un objeto distribuido por objetos que residen en diferentes máquinas en entornos heterogéneos

- Los objetos pueden estar desarrollados en diferentes lenguajes.
- Los equipos pueden tener diferente:
  - Hardware
  - Sistema operativo
- Los equipos pueden estar conectados entre sí usando distintos protocolos de comunicación

Su objetivo es facilitar el desarrollo de aplicaciones distribuidas

# CORBA

## Propiedades deseadas

### Transparencia de distribución

- Ni cliente ni servidor necesitan saber si la aplicación está distribuida o centralizado

### Transparencia de localización (caso distribuido)

- El cliente no necesita saber donde se encuentra el objeto de implementación
- El servicio que presta el objeto de implementación no necesita saber donde se ejecuta el cliente

### Integración de software existente

- Amortizar inversión previa sistemas heredados (legacy systems)

# CORBA

## Propiedades deseadas

### Activación de objetos

- Objetos remotos no tienen que estar en memoria permanentemente
- Se hace de manera invisible para el cliente

### Comunicación flexible

Múltiples modos de comunicación

Síncrona: tipo invocación de métodos / RPC

Asíncrona: sin respuesta

Múltiples modelos de comunicación

Invocación estática

Invocación dinámica

# CORBA

## Historia.

❖ OMG (Object Management Group)

- Consorcio creado en 1989, primer "producto": CORBA
  - inicialmente 8 empresas (Sun, HP, 3Com,...)
    - Hoy: más de 800 socios
    - Proveedores de SW y equipos, operadores de telecomunicaciones, empresas, universidades,...

**OMG** → **Proporcionar interoperabilidad entre aplicaciones en un entorno distribuido y heterogéneo.**

# CORBA

## Historia

La OMG Object Management Archictecture pretende reducir la complejidad, los costos y acelerar la introducción del sw: Una visión de componentes sw trabajando juntos, sin preocuparse de detalles tales como:

- Ubicación de sus componentes
- El sistema operativo
- Lenguajes de programación
- Software y hardware de la red(protocolos y plataformas)

Producto inicial: 1990 **OMA (Object Management Architecture)**
**CORBA (Common Object Request Broker Arquitecture)**

- CORBA 1. 1991
- CORBA 2. 1995
- CORBA 3. 2002

# CORBA

OMG solo define especificaciones, no realiza ninguna implementación de ellas, son las empresas y/o grupos de desarrolladores los que se deben ocupar de esto

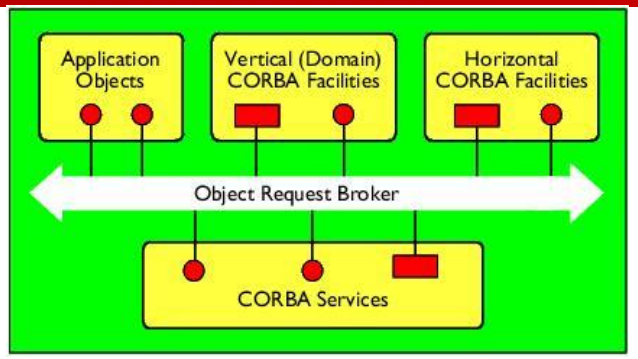
**Proceso de adopción**

* Petición de propuestas(RFP).Cerrada.Iniciativa del OMG. (recolección)
* Petición de comentarios(RFC).Iniciativa de un socio.
* Petición de información (RFI). Abierta.Prospectiva(sw disponible?).

**RFP**

* Propuesta por una TF
* Cartas de intención
* Propuestas iniciales
* Propuesta revisada
* Evaluación y recomendación por parte de la TF
* En total 12-15 meses

# CORBA




## OMA (Object Management Architecture)

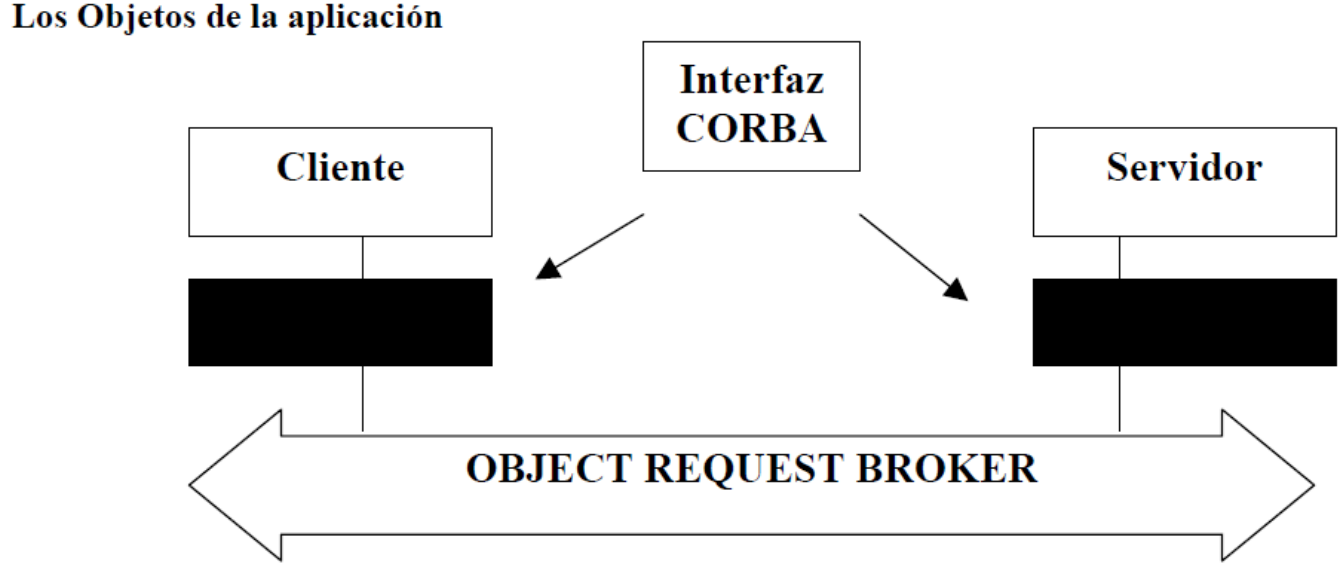
- En 1990 la OMG pública la primera especificación de OMA.
- Arquitectura a partir de la cual se pueden construir las aplicaciones distribuidas y de la que CORBA es parte.
- Esta arquitectura esta compuesta por cuatro principales elementos:

  - ORB (Object Request Broker): define el bus de objetos CORBA
  - CORBAServices: definen el framework de objetos a nivel del sistema
  - CORBAFacilites: definen el framework de aplicaciones horizontales y verticales que son usadas por los objetos de negocios.
  - Objetos de aplicación: Son los objetos de negocios y de aplicación.

# ELEMENTOS DE LA ARQUITECTURA OMA

## ORB

**Object Request Broker**
**Intermediario de petición de Objetos**

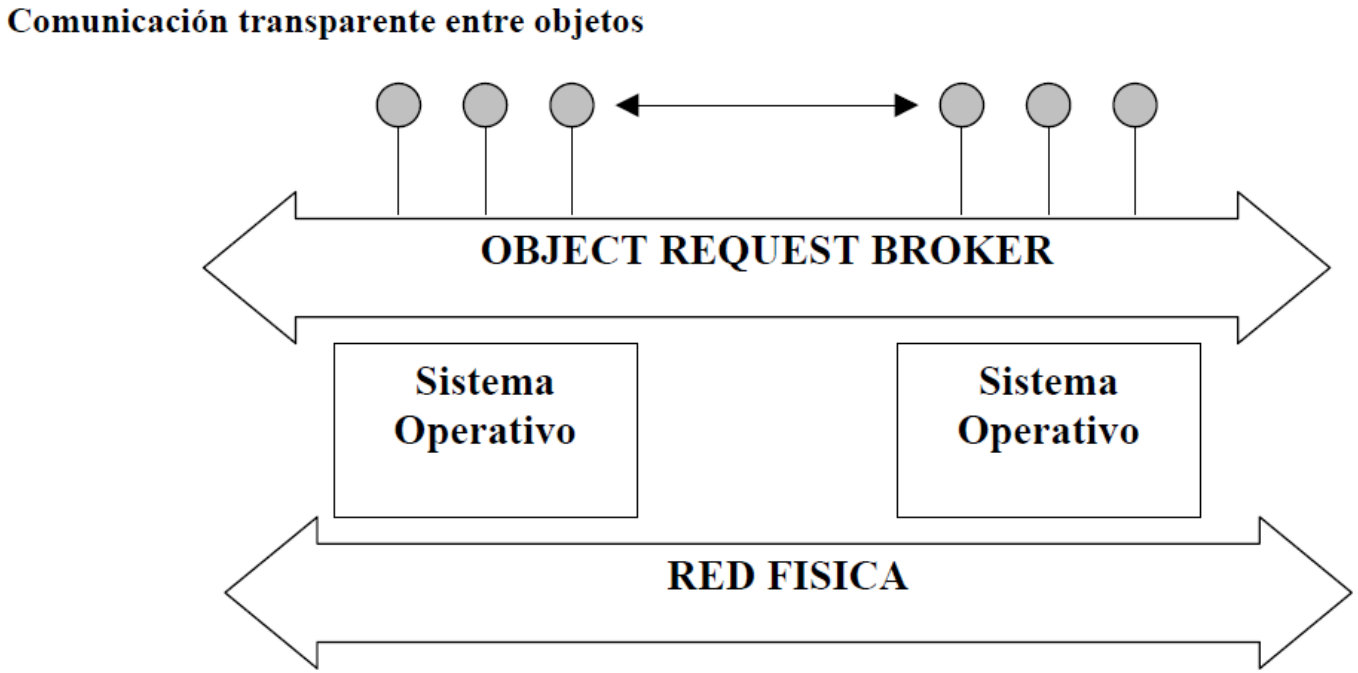



- Los objetos específicos de la aplicación se conectan al ORB por interfaces especificadas en IDL pero no tienen ninguna otra restricción (cualquier lenguaje, heredados ...)

# ELEMENTOS DE LA ARQUITECTURA OMA

## ORB

**Object Request Broker**
**Intermediario de petición de Objetos**



# ELEMENTOS DE LA ARQUITECTURA OMA

## ORB

Núcleo de CORBA: Un bus de software

- Oculta la heterogeneidad a las dos partes
- Oculta los detalles de la comunicación a las dos partes

Proporciona transparencia de localización via referencias de objeto

- Desconocimiento de la residencia de los objetos
- Interpreta las referencias de objeto
- Canaliza las invocaciones del cliente al objeto remoto correcto
- Canaliza las respuestas del objeto remoto al cliente

# ELEMENTOS DE OMA

## ORB

**Proporciona transparencia de implementación**

- El cliente no conoce el lenguaje de implementación de los objetos corba

**Proporciona transparencia de distribución**

- Comportamiento igual en centralizado o distribuido

**Proporciona transparencia de estado de ejecución**

- Se ocupa de la activación y desactivación de objetos

**Proporciona servicios para construir peticiones dinámicamente**

- Construye en tiempo de ejecución el stub y skeleton.

**Proporciona transparencia de mecanismos de comunicación**

- Se ocupa de abstraer los protocolos de red utilizados en la comunicación.

# ELEMENTOS DE OMA

## SERVICIOS CORBA

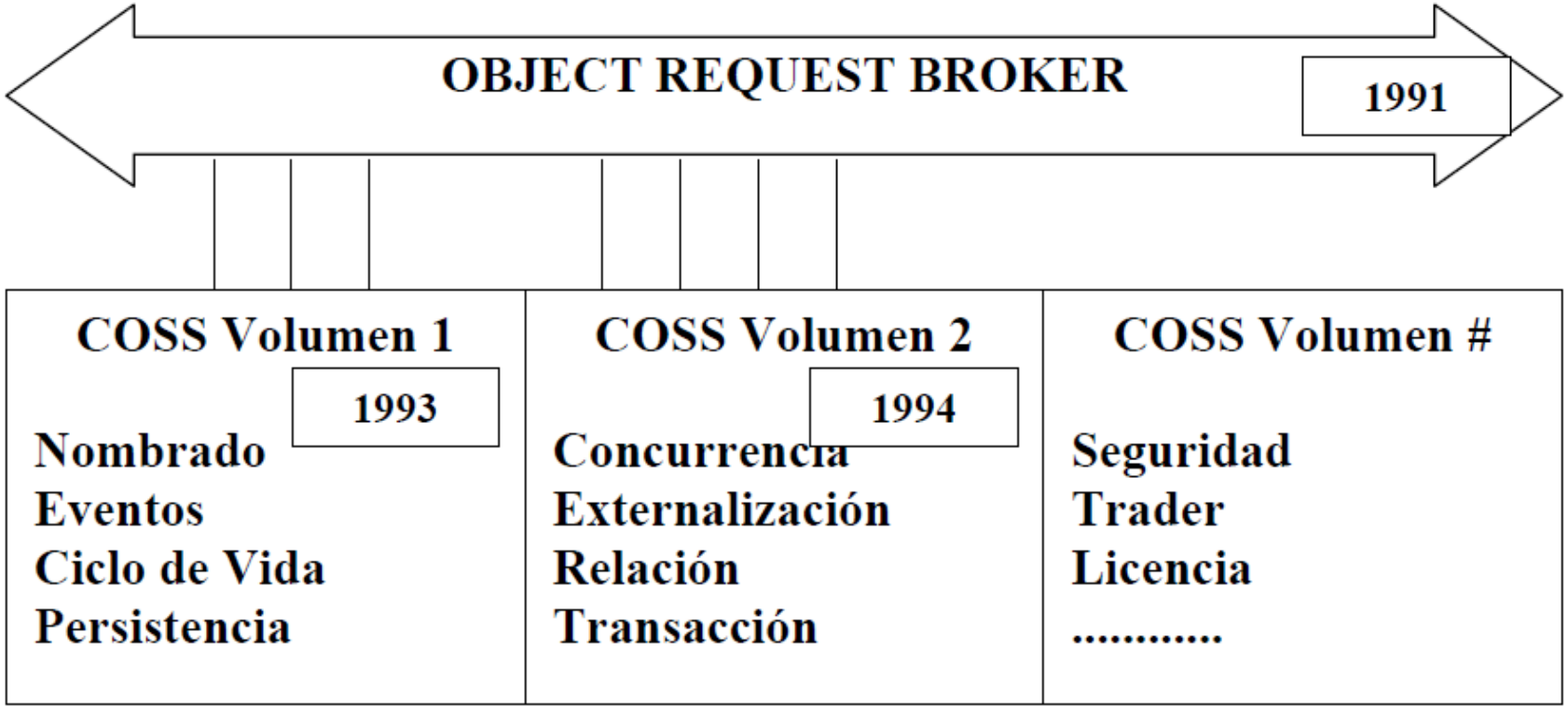
- Servicios fundamentales y de uso frecuente en diversas aplicaciones.
- Son considerados Servicios de Bajo Nivel (en contraste con las facilidades comunes).
- La colección de estos servicios OMG se les ha llamado CORBAServices.

Los servicios

* Estandarizan la gestión del ciclo de vida de los objetos
* Habilitan la creación de objetos
* Controlan el acceso a los objetos
* Mantienen las reglas de relación entre objetos
* Dan un entorno genérico de ejecución

# ELEMENTOS DE OMA

## SERVICIOS CORBA



# ELEMENTOS DE OMA

## SERVICIOS CORBA

### Servicio de nombrado

- Es un mecanismo para que los objetos localicen otros objetos.
- La creación de una asociación nombre – referencia a objeto se llama name binding
- Un contexto de nombres es el ámbito en el cual las asociaciones son únicas (se garantiza id únicos aunque un objeto puede tener varios nombres).
- Puede tener jerarquías (directorios) sobre las cuales el cliente puede navegar para resolver un nombre.

# ELEMENTOS DE OMA

## SERVICIOS CORBA

### Servicio de eventos

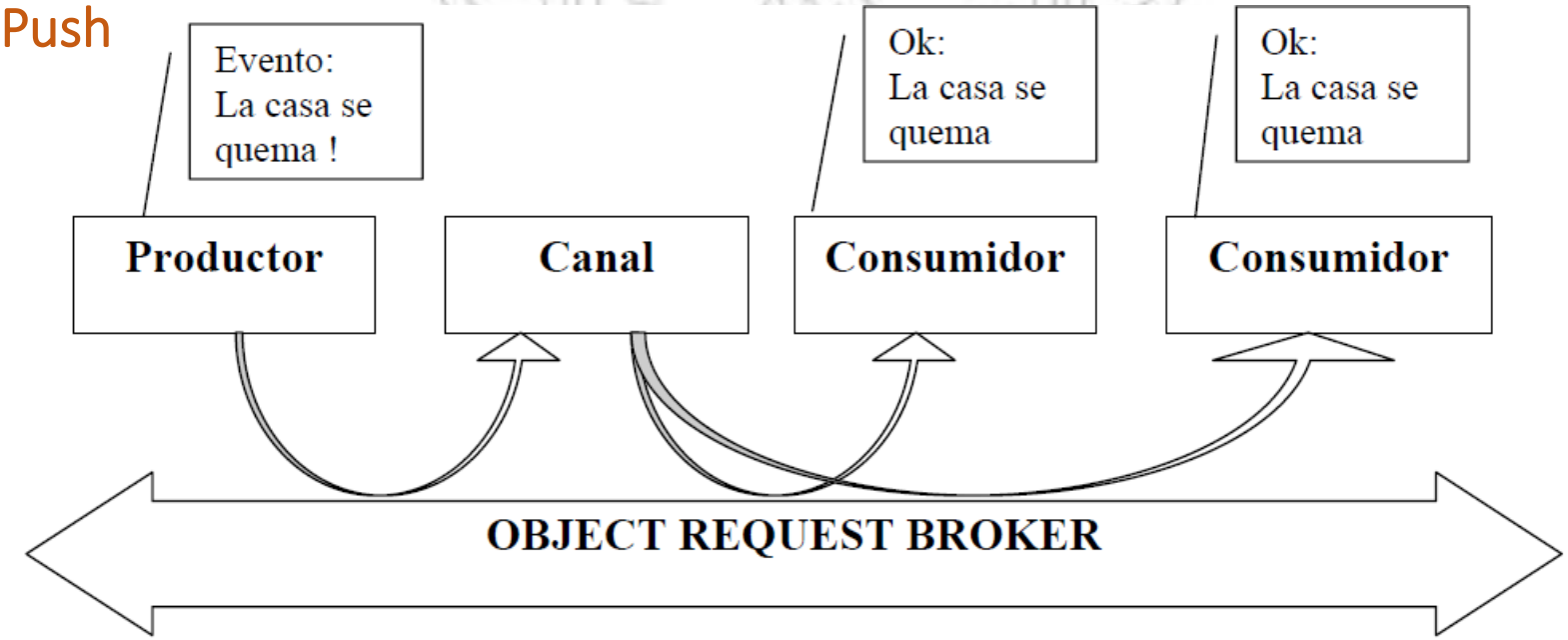
- Permite que los objetos se registren dinámicamente para recibir eventos que se producen en un objeto de su interés.
- Los eventos pueden tener tipo, el cual sirve de mecanismo de filtrado.
- Se distinguen dos roles: consumidor o productor.
- Una notificación es un mensaje (normalmente oneway) que un objeto envía para indicar que se produjo un evento.

# ELEMENTOS DE OMA

SERVICIOS CORBA

Servicio de eventos

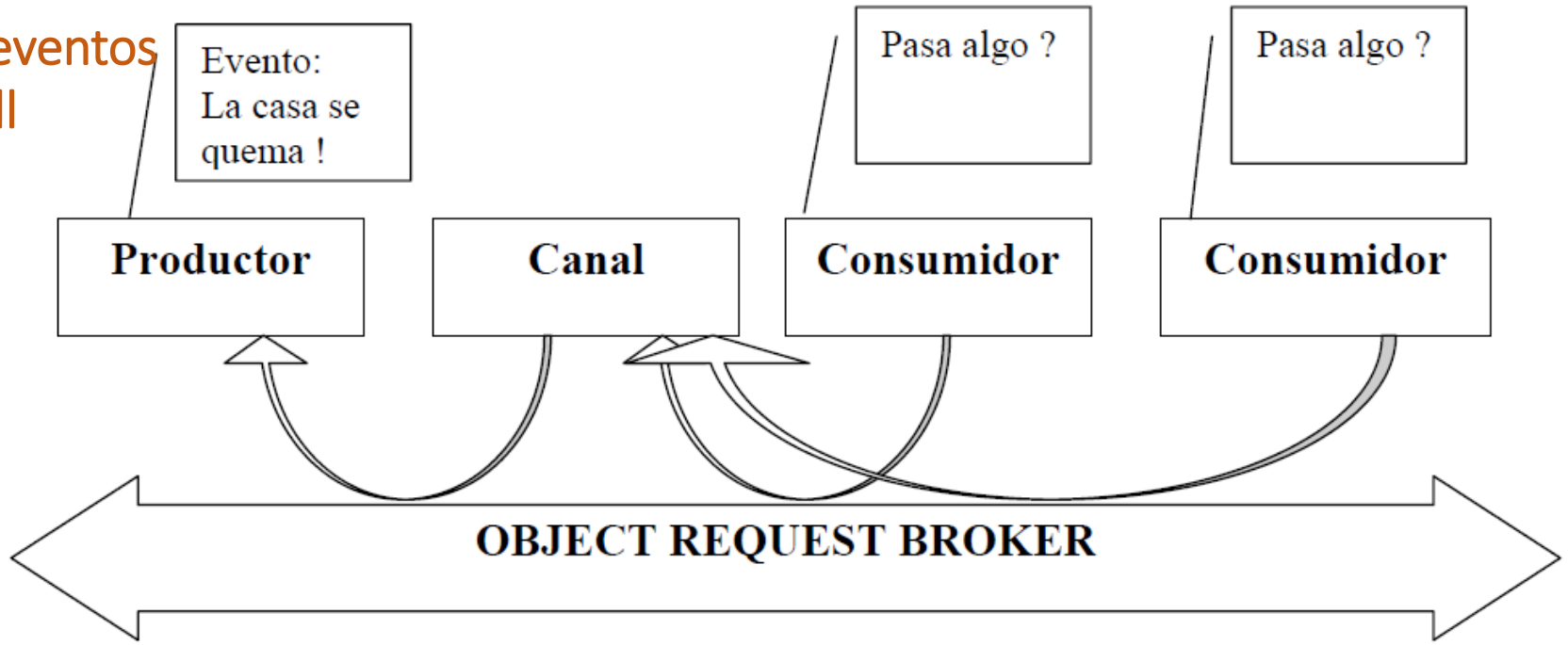
Modelo Push



# ELEMENTOS DE OMA

## SERVICIOS CORBA

### Servicio de eventos
### Modelo Pull



# ELEMENTOS DE OMA

## SERVICIOS CORBA

### Servicio de seguridad

- Servicio de Manejo de lista de control de acceso.
- Permite autenticación para usuarios y servidores.
- Permite la generación y delegación de credenciales.
- Auditoria por los servidores sobre la invocación de métodos remotos.
- Antes de hacer uso de un servicio el cliente debe enviar sus credenciales en un mensaje. Estas credenciales son válidas en el servidor.

# ELEMENTOS DE OMA

## OTROS SERVICIOS CORBA

### COSS I

* Persistent Object Service: existencia a largo plazo de objetos, gestión de almacenamiento de objetos.

### COSS II

* Relationship Services: gestión de representación y consistencia de relaciones entre objetos
* Externalization Services: capacidad para almacenar objetos (->re-internalización)
* Transaction Services: mezcla de OLTP y objetos distribuidos
* Concurrency Control Service: gestión de la ejecución concurrente

# ELEMENTOS DE OMA

## OTROS SERVICIOS CORBA

### COSS III & COSS IV

- **Query Services:** Permite a usuarios y objetos invocar consultas en colecciones de otros objetos
- **Property Services:** Manipulación de propiedades de objetos
- **Licencing Services:** Control de propiedad intelectual
- **Security Services:** Identificación, autenticación
- **Time Services:** Hora y temporizadores
- **Trade Service:** Localización de objetos suministrando información del servicio requerido.

# ELEMENTOS DE OMA

## Facilidades Comunes

- Son funciones de aplicación genérica que pueden ser configuradas para una necesidad específica. La OMG las adopta como CORBAFacilities
- Están definidas en términos de interfaces
- Se configuran de acuerdo a la aplicación y se sitúan más cerca del usuario (entorno de trabajo)

**Tipos**

Independientes del dominio de aplicación

**Dependientes del dominio de la aplicación**

# ELEMENTOS DE OMA

## Facilidades Comunes

### Horizontales:

Se agrupan en cuatro conjuntos principales

- Interfaz de usuario: Gestión de representación de documentos compuestos
- Gestión de información: Almacenamiento y recuperación de información
- Gestión de sistemas. Herramientas de gestión
- Gestión de tareas: Automatización de trabajos.

# ELEMENTOS DE OMA

## Facilidades Comunes

Proveen funcionalidad de interés para usuarios finales en dominios de aplicación particulares.

- Telecomunicaciones
- Finanzas
- Medicina
- Transporte
- Comercio electrónico

# ELEMENTOS DE OMA

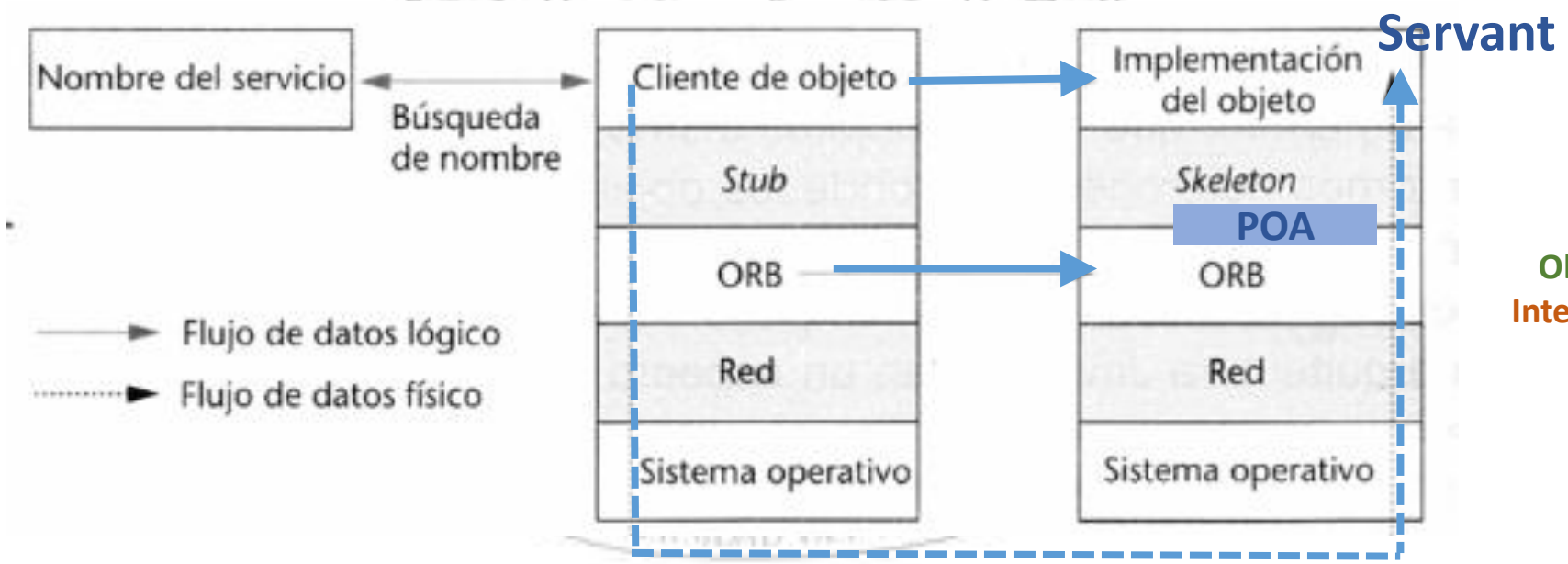
## Objetos de aplicación

Los objetos específicos de la aplicación se conectan al ORB por interfaces específicas en IDL pero no tienen ninguna otra restricción.

- ❖ No estandarizados por la OMG
- ❖ Objetos que efectúan tareas especificas a usuarios
- ❖ Construidas a través de un conjunto de objetos:

- CORBAFacilities
- CORBAServices

# ARQUITECTURA BASE DE CORBA




**Object Request Broker**
**Intermediario de petición de Objetos**

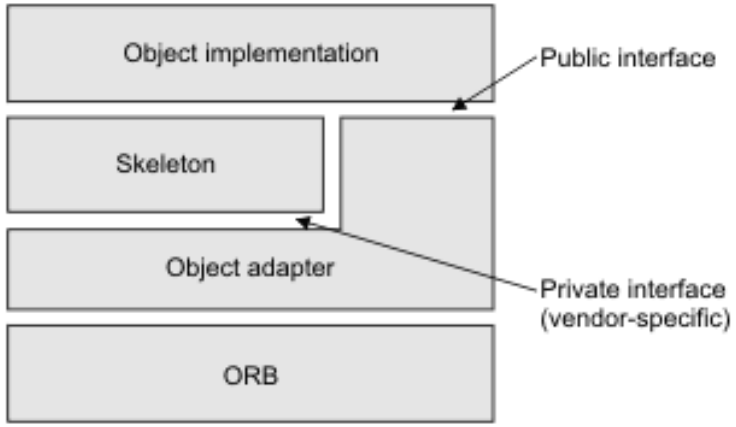
Servant: Entidad desarrollada en un lenguaje que implementa las operaciones definidas en la interface.

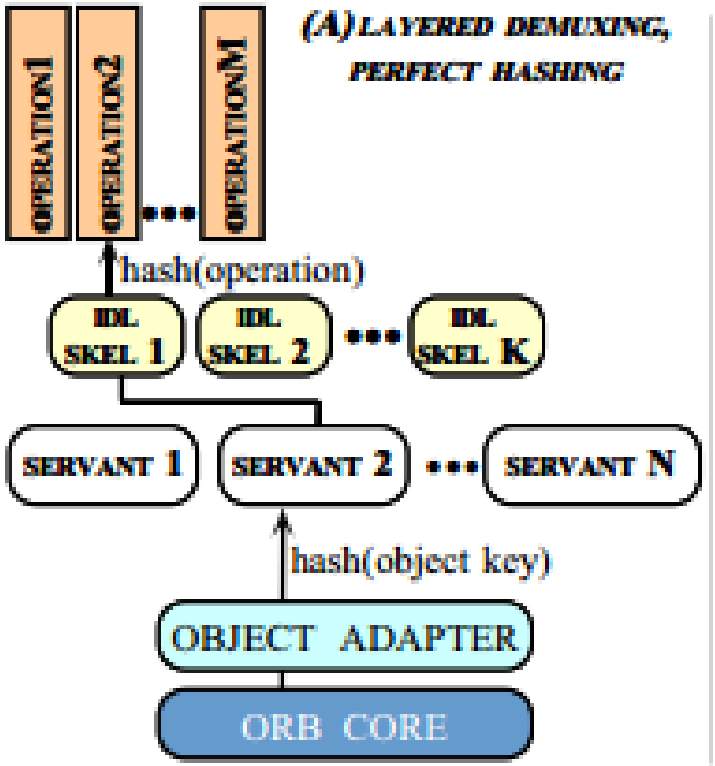
Stub: Crea y emite los pedidos en nombre del cliente al ORB

Skeleton: entrega el pedido a la implementación del objeto CORBA

# ARQUITECTURA BASE DE CORBA

Desmultiplexacion de una petición en CORBA por medio del adaptador de objetos.



# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## La interface de objetos de CORBA

Un objeto distribuido se define por medio de una interface escrita en CORBA IDL (sintaxis parecida a c++ o java)


- La definición de la interfaz IDL es independiente del lenguaje de programación, es procesada por un compilador.
- Cualquier cliente que desea invocar una operación en el objeto corba, debe utilizar una interfaz IDL para especificar la operación que quiere llevar a cabo
- Un objeto definido en CORBA IDL puede implementarse en un gran número de lenguajes.

# ELEMENTOS DE LA ARQUITECTURA DE CORBA

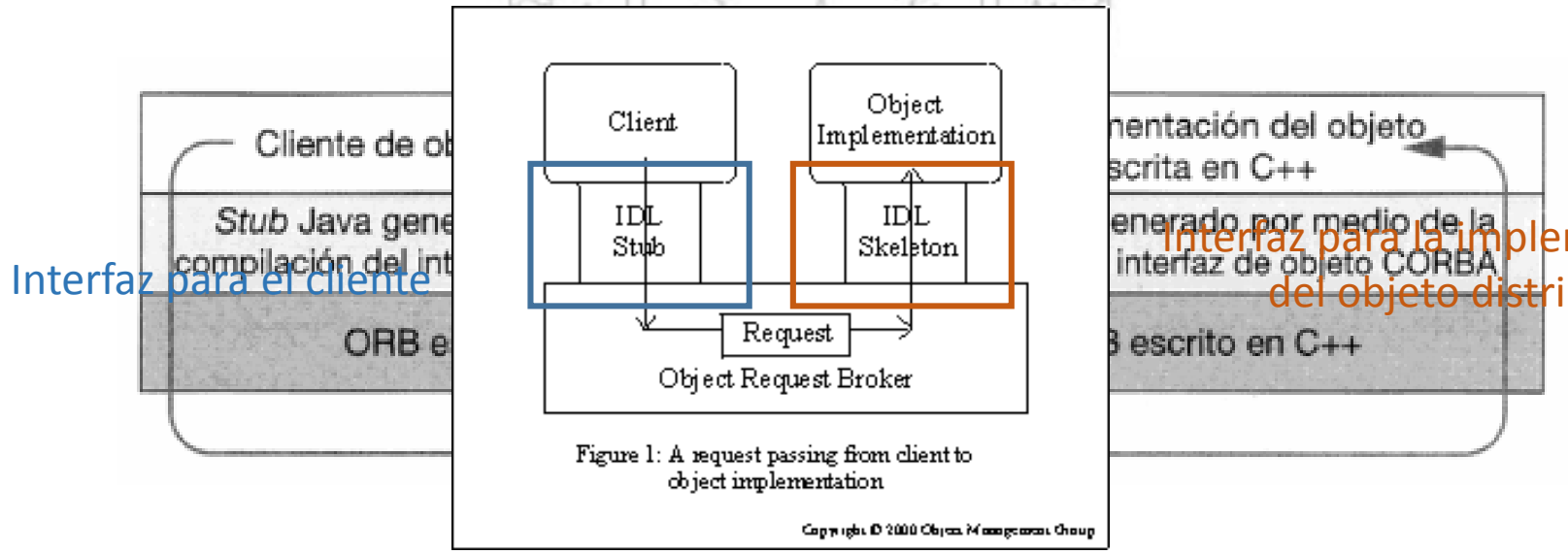
## Que es un objeto CORBA

- En el modelo CORBA, un objeto CORBA es un objeto virtual es decir no tiene existencia por sí mismo si no hay detrás un servant en un determinado lenguaje de programación que le soporte.
- Esto permite que cliente y servidor puedan escribirse en diferentes lenguajes de Programación
- Un objeto corba puede referenciar a una aplicación desarrollada en un lenguaje que no sea orientado a objetos.

# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## La interface de objetos de CORBA

- La separación de interface e implementación es fundamental para lograr la interoperabilidad.
- Los clientes acceden a los objetos sólo a través de su interfaz, invocando sólo aquellas operaciones que el objeto expone a través de su interfaz IDL.

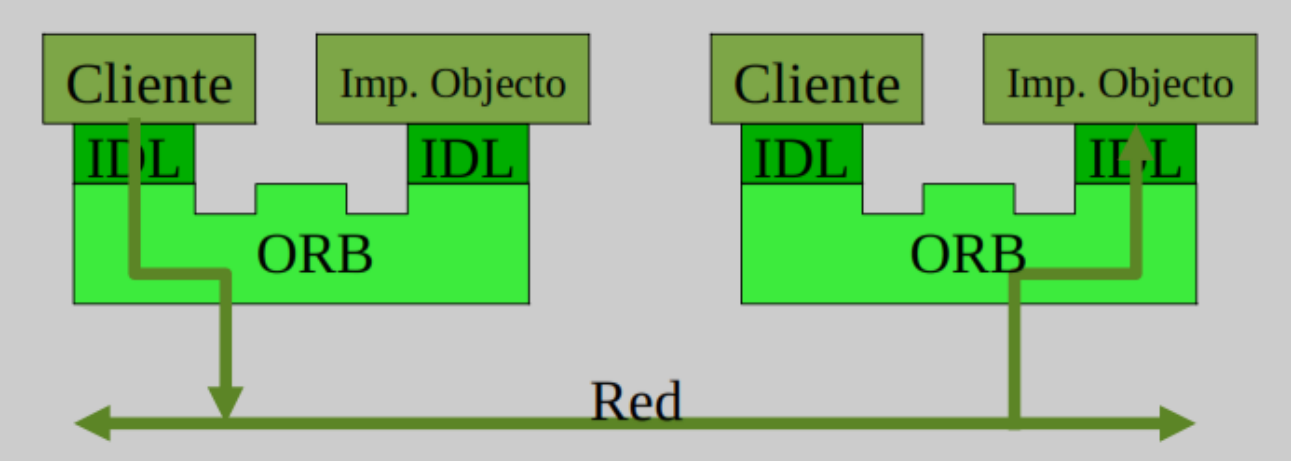


# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## ORB

- La inter-operabilidad es un asunto entre vendedores de ORB's y no entre vendedores y usuarios operaciones y tipos.

- Un cliente no hace nada distinto a una invocación de un objeto.




A pesar de estar escritos en diferentes lenguajes, los ORB pueden interactuar debido al uso de protocolos comunes soportados por ambos.

# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## ORB


- Pueden encontrarse en diferentes dominios.
- Pueden interoperar por medio de ORBs que hacen de switch entre comunicaciones.

# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## Protocolos utilizados entre ORB

Diferentes ORB utilizan un protocol comun

### General Inter-ORB Protocol (GIOP)

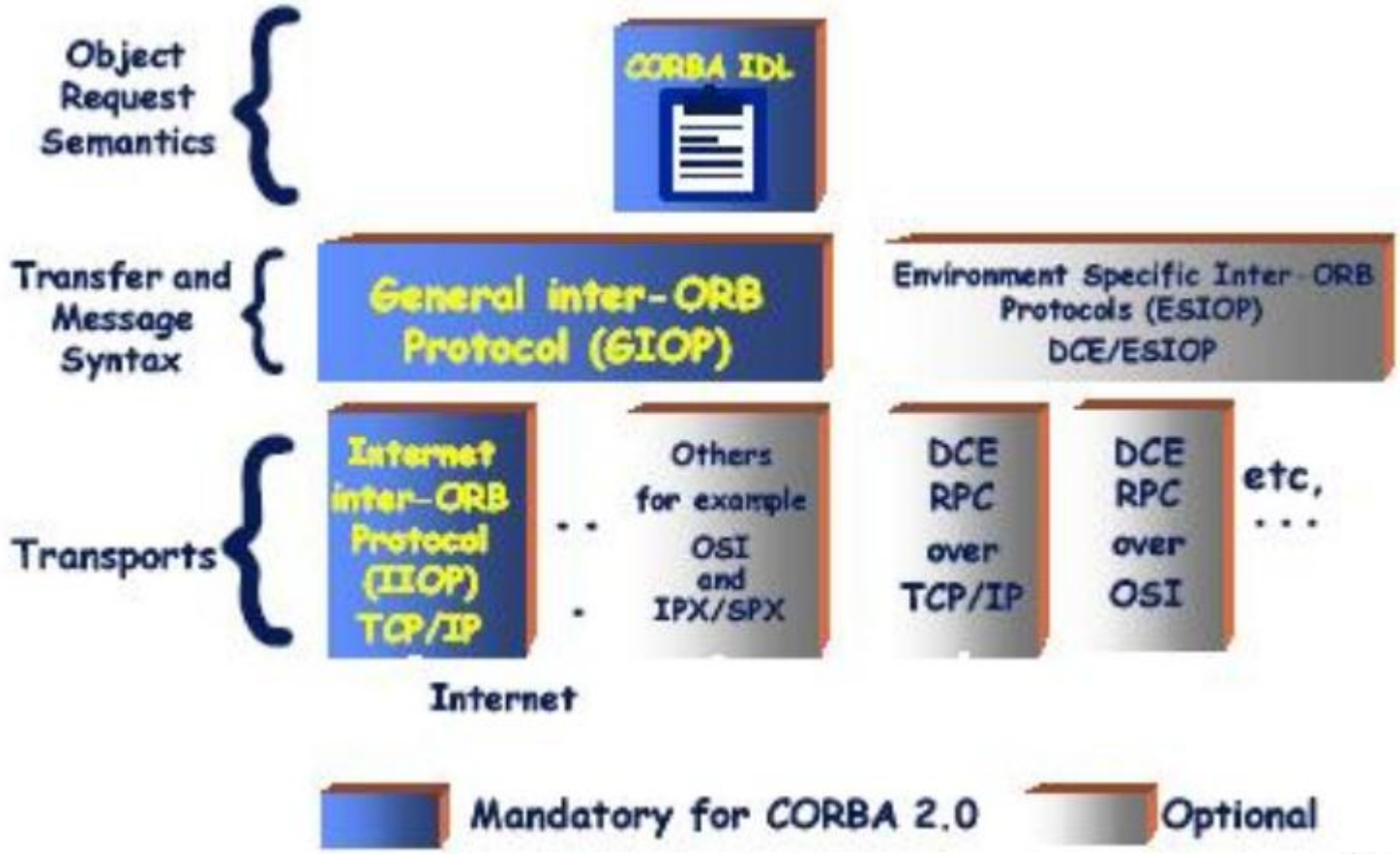
- GIOP es un protocolo estándar de alto nivel para la comunicación entre ORBs.
- GIOP es un protocolo generalizado, no se usa directamente, sino que es especializado por un protocolo particular para poder usarse

### Internet Inter-ORB Protocol (IIOP)

- Una especialización del GIOP
- Protocolo estándar para la comunicación entre ORBs en redes basadas en TCP/IP.
- Un ORB debe soportar IIOP para ser considerado CORBA 2.0 compatible

# ELEMENTOS DE LA ARQUITECTURA DE CORBA

## Protocolos utilizados entre ORB



# ¿Como localizar un objeto CORBA?

## Referencia a objetos corba

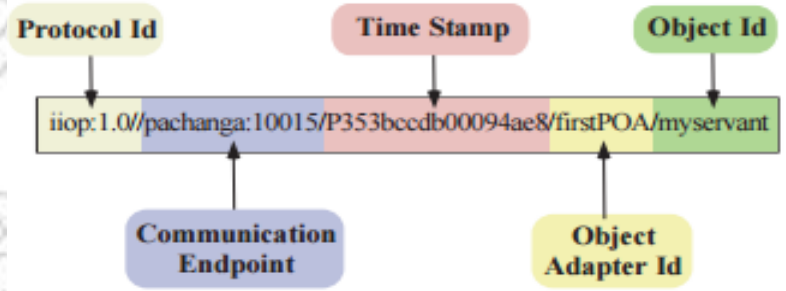
- Una instancia de un objeto corba es representado por el IOR (Interoperable Object Reference)
- CORBA 2: Formato estándar de una referencia a un objeto remoto (es opaco para el programador)
  - Para evitar conflictos entre implementaciones de diferentes vendedores
- La refeferencia a un objeto corba encapsula
  - Dirección de red del proceso servidor
  - Un identificador único (puesto por el servidor) que identifica la implementación concreta a la que va dirigida la petición

# ¿Como localizar un objeto CORBA?

## Referencia a objetos corba

*Formato para las referencias de objetos remotos según la especificación CORBA 2.0, denominado IOR (Interoperable Object Reference).*

| Nombre de tipo de interfaz IDL | Protocolo y dirección detallada | | | Clave de objeto | |
|---|---|---|---|---|---|
| Identificador de repositorio de interfaz | IIOP | Nombre de dominio del host | Número de puerto | Nombre de adaptador | Nombre de objeto |




1) Especifica el nombre de tipo de la interfaz IDL del objeto CORBA. Si existe un repositorio de interfaces corresponde al identificador de la interface en el repositorio.

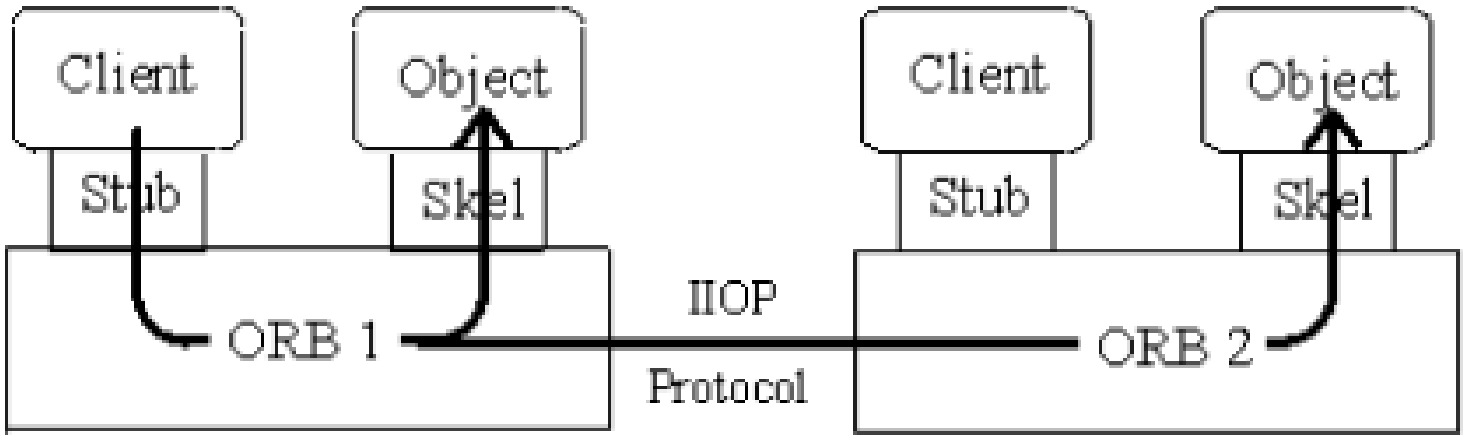
2) Protocolo de transporte y los detalles para identificar al servidor.

3) Permite identificar al objeto corba. Se establece el nombre del adaptador de objetos y el nombre del objeto corba en el adaptador de objetos.

# ¿Como localizar un objeto CORBA?

## Referencia a objetos corba

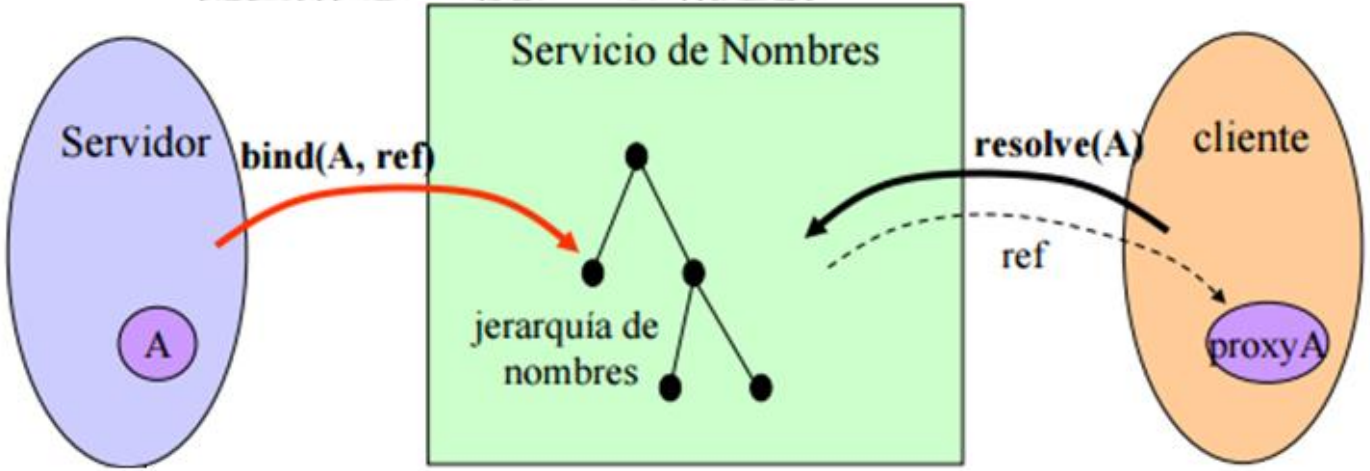
- Para poder invocar una operación de la instancia de un objeto remoto, el cliente debe primero obtener la referencia de objeto.
- Cuando el ORB examina la referencia al objeto y descubre que el objeto de destino es remoto, enruta la invocación por la red al ORB correspondiente al objeto remoto.



# ¿Como localizar un objeto CORBA?

## Servidor de nombres

- En CORBA, el servidor de nombres es global y puede estar en cualquier máquina ( puede escuchar por cualquier puerto) hay que localizarlo, solicitándoselo al ORB.
- El Servicio de Nombres guarda pares <nombre, referencia de objeto>
  - Los nombres están organizados en una jerarquía




(*) Gráfico: Juan Pavón. UCM

# ¿Como localizar un objeto CORBA?

## Servidor de nombres

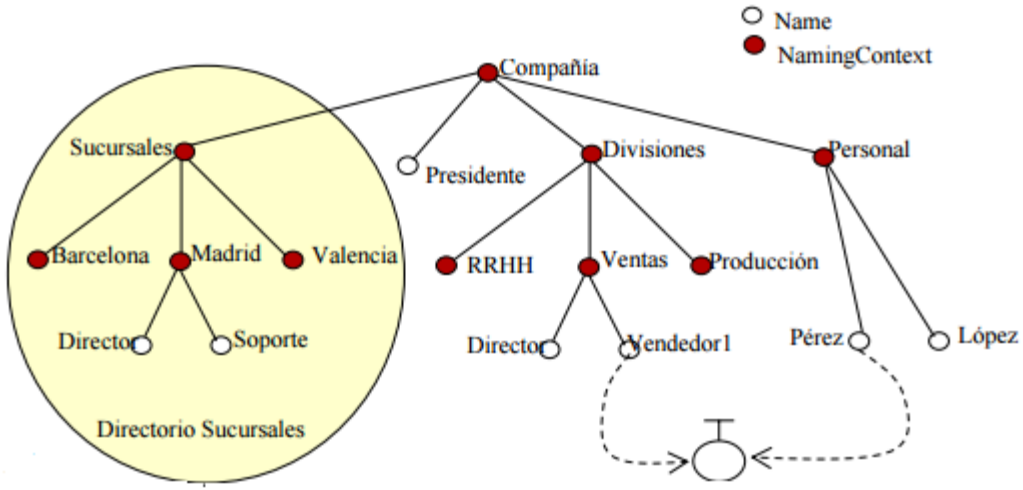
Los nombres en el Servicio de Nombres se organizan jerárquicamente (como sist de ficheros)

Cada nodo en la jerarquía de nombres puede ser:

**Contexto de nombrado:** Define un espacio de nombres

**Name:** Tiene asociada una referencia a un objeto

**El nombre compuesto:** Nombre del objeto y sus Contextos de nombrado.




(*) Gráfico: Juan Pavón. UCM

# CORBA

## MODOS DE COMUNICACIÓN

### SINCRONA

- Invocación de métodos remotos normal.
- Repuesta puede ser de tipo void

### ASINCRONA

- Invocación de métodos definidos como oneway

  Un cliente invoca una operación, en la cual un cliente no se bloqua mientras se ejecuta.

# CORBA

## Semántica de entrega de mensajes

- Si la comunicación es síncrona es **exactamente una en ausencia de errores.** En el caso de que se produzca un error se asegura la semántica **como mucho una.**

- Se permite comunicación no bloqueante de dos tipos:
  - Asíncrona
  - En un solo sentido.

- En las operaciones **asíncronas se asegura la semántica como mucho una** y en las operaciones en las que el cliente no espera respuesta del servidor, es decir en un solo sentido, **la semántica es *best-effort*.**

# CORBA

## Como funciona

- **El servidor**
  - Crea objetos remotos
  - Hace accesibles refs a objetos remotos
  - Espera a que los clientes invoquen a estos objetos remotos o a sus métodos
- **El cliente**
  - Obtiene una referencia de uno o más objetos remotos en el servidor
  - Invoca a sus métodos

# CORBA

Para cada lenguaje de programación existe un estandar OMG que especifica como mapear los tipos e invocaciones para convertirlos en tipos y funciones en un lenguaje.

**Las invocaciones pueden ser: dinámicas o estáticas.**

- **Invocaciones estáticas:** Son encaminadas al ORB del cliente vía el stub.
- **Invocaciones dinámicas:** Son ensambladas en tiempo de ejecución por el cliente. (DII proporcionan una gran flexibilidad).

# CORBA

## Invocación estática – Stub IDL

- Interfaz que ve el cliente ( respecto de los servicios del servidor). Actúa como un representante del servidor en el lado del cliente (proxy) que le produce la impresión de una invocación local aunque sea remota.
- Se realiza la invocación remota y el marshalling de los datos de los parámetros de la operación
- Se genera a partir de la especificación IDL
- Los tipos de objetos con los cuales se puede interactuar se conocen en tiempo de compilación

# CORBA

## Invocación estática – Stub IDL

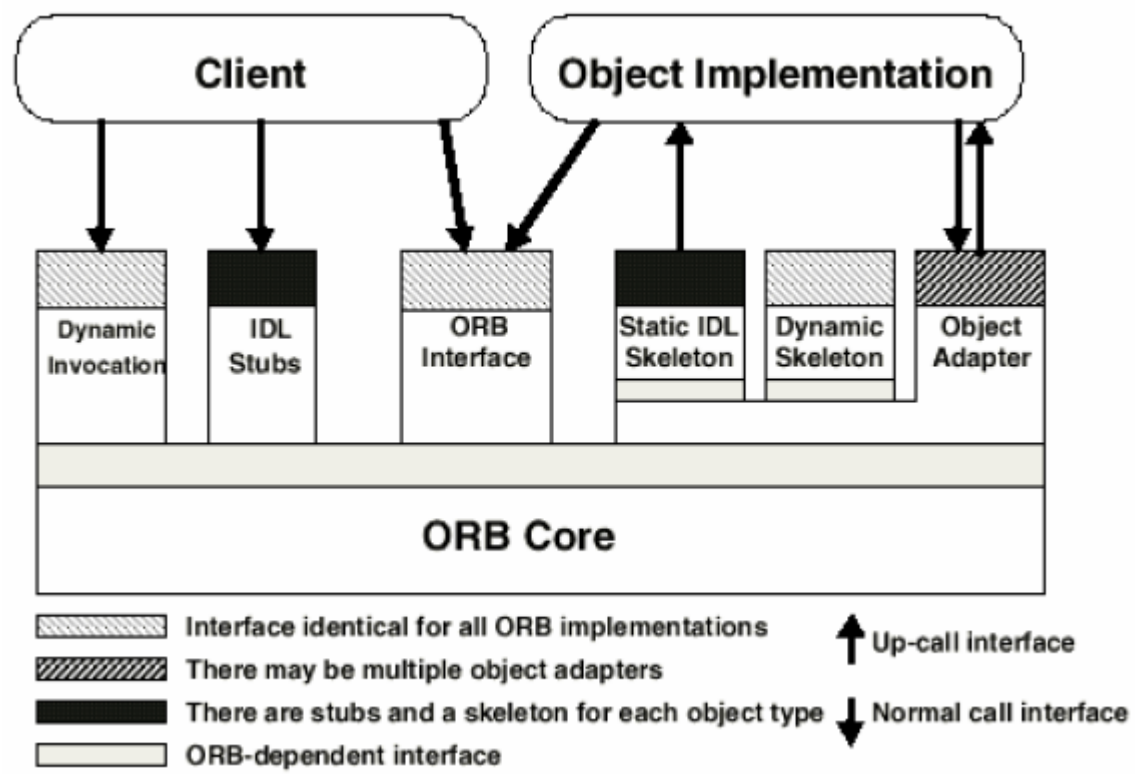
El cliente conoce el objeto a ser invocado, y el stub y skeleton son generados por el IDL.

Lo anterior significa que el cliente conoce:

- Las operaciones que puede invocar del objeto.
- Cuales son los parametros de entrada, su orden y el tipo de resultado de las operaciones.

# CORBA

## Elementos de la arquitectura



# CORBA

## ¿Cómo se realiza la invocación?

- El cliente
  - ¿Qué necesita?
    - Referencia del objeto remoto IOR
    - Tipo del objeto
    - Nombre de la operación que se desea invocar
  - ¿Cómo realiza la invocación?
    - Usando stub generado a partir del IDL
    - Usando invocación dinámica a través de DII
  - Tiene que gestionar las excepciones producidas por su llamada
- El ORB a partir de la petición del cliente
  - Encuentra el código de la implementación apropiada,
  - Transmite los parámetros
  - Transfiere el control a la Implementación de la Interfaz a través de:
    - esqueleto IDL,
    - esqueleto dinámico (DII)
  - Informa de excepciones en el proceso (ej referencias IOR o IDL no válidas)

# CORBA

## ¿Cómo se realiza la invocación?

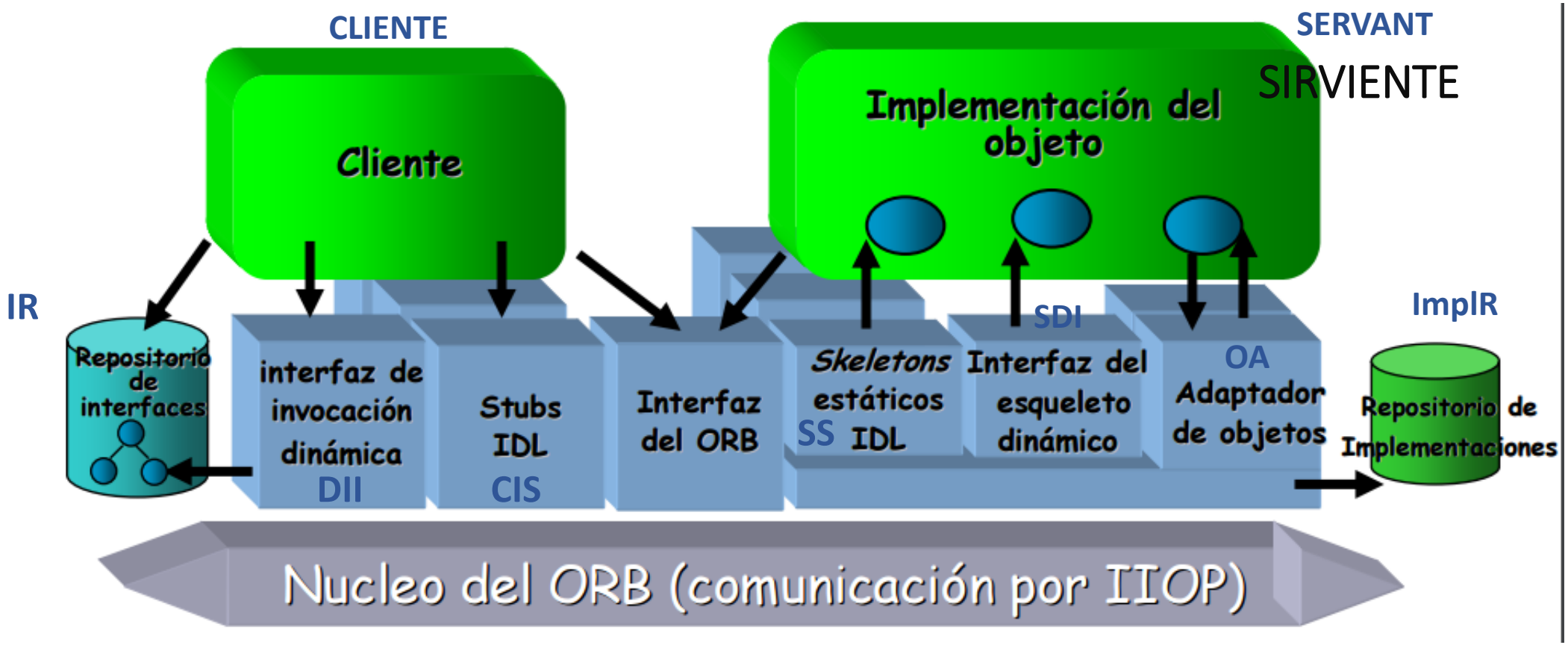
- Para el desarrollador es una invocación corriente
- En tiempo de ejecución la invocación pasa por:
  - El stub del cliente
  - El ORB
  - El adaptador de objetos
  - Encontrar el objeto adecuado
  - Viajar por los skeletons del servidor
  - Realizar la invocación sobre el objeto
  - Realizar el mismo camino de vuelta

# CORBA

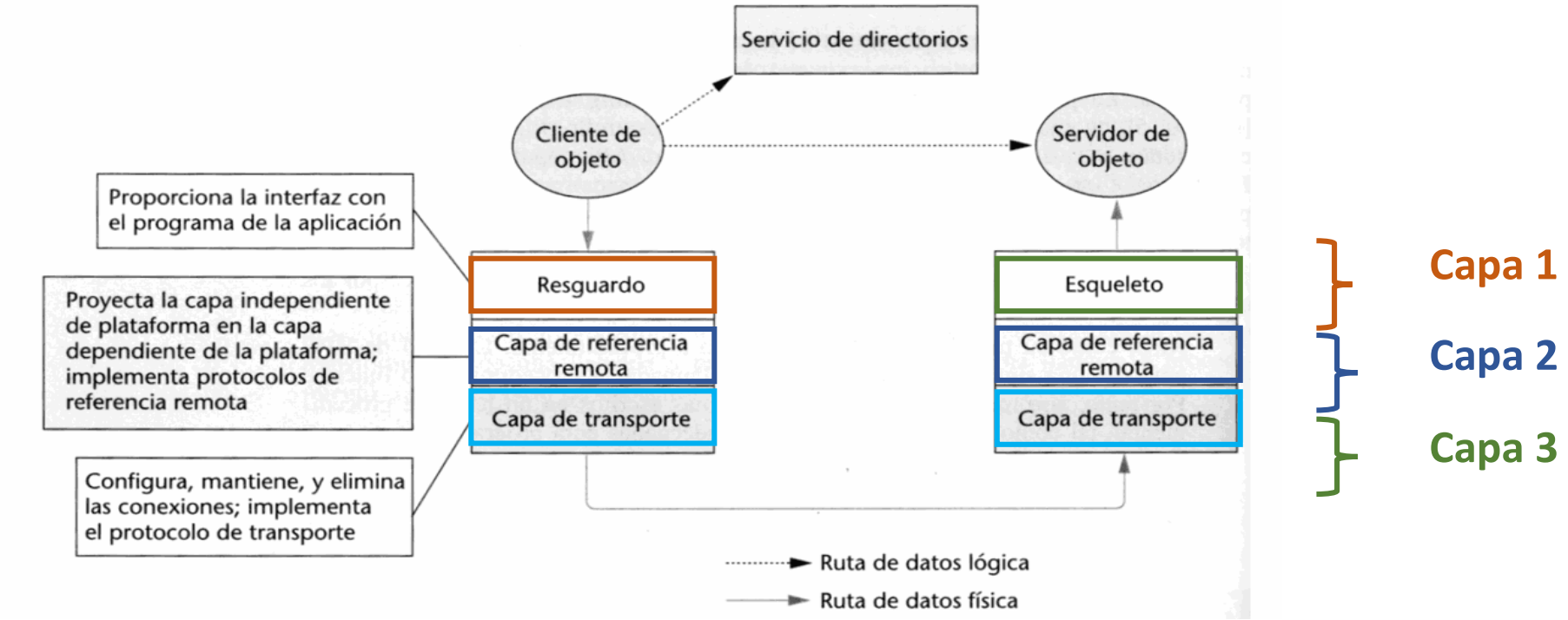
## ¿Cómo se recibe la petición?

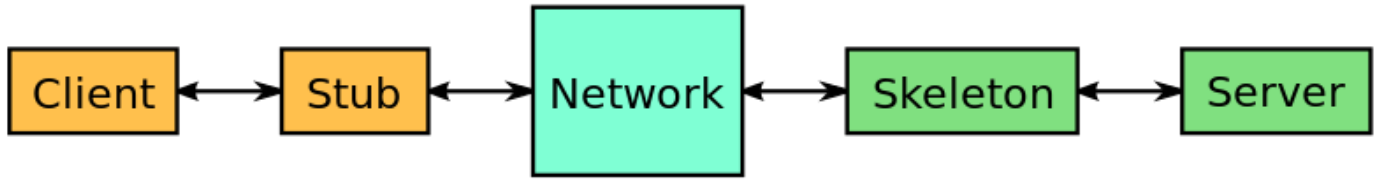
- La implementación del objeto que proporciona el servicio
  - Recibe la invocación de uno de sus métodos como llamadas desde el ORB hacia la Implementación de la Interfaz.
    - La llamada puede venir de un cliente :
      - que ha utilizado los stubs IDL,
      - Que ha utilizado DII
    - Los esqueletos de la interfaz IDL son específicos de cada interfaz y del adaptador de objetos que exista en la implementación de CORBA.
  - Cuando la invocación ha sido completada, el control y los valores de retorno son devueltos al cliente.
  - Puede utilizar los servicios que proporciona el adaptador de objetos e incluso los que proporciona el ORB, mientras procesa la petición que ha recibido del cliente
    - Puede elegir un Adaptador de Objetos entre un conjunto de ellos, una decisión que estará basada en la clase de servicios que pueda requerir la Implementación de la Interfaz

# Arquitectura de CORBA



# Comparación JAVA RMI vs Componentes de CORBA



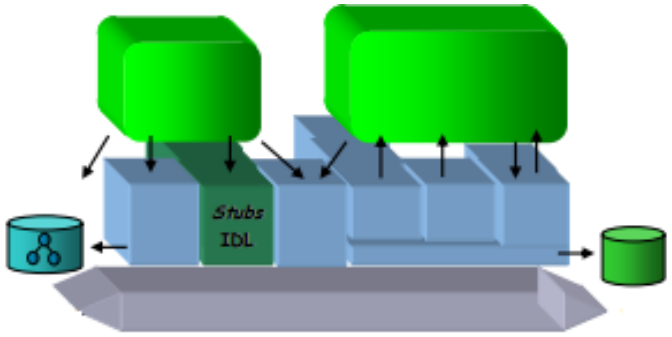



# Arquitectura de CORBA

## Lado cliente (1/5)

### Stub IDL

- Interfaz estática a los servicios declarados en las interfaces IDL
- Permite que para el cliente todas las llamadas parezcan locales.
- Actúa como proxy (representante) del objeto remoto
  - La invocación de métodos remotos, incluyendo el marshalling
  - Recepción de respuestas, incluyendo el unmarshalling
- El cliente puede tener tantos stubs como interfaces IDL existan
- Generado a partir del IDL en el lenguaje de programación del cliente (C, C++, Java, Smalltalk, ADA,… ) por un compilador IDL.

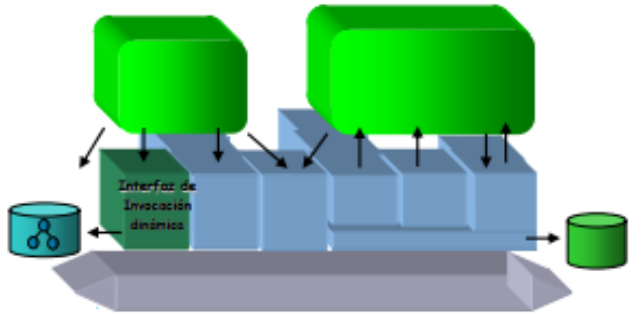


# Arquitectura de CORBA

## Lado cliente (2/5)

### Interfaz de Invocación Dinámica (DII)

- Permite descubrir en run-time las interfaces de Objetos CORBA que no poseen stubs estáticos.
- CORBA define API's para buscar los métodos y sus parámetros de una interfaz (“metadatos”).
- Utilizado para hacer una invocación dinámica después de construida mediante la interfaz del ORB
- El ORB hace el marshalling, no la DII

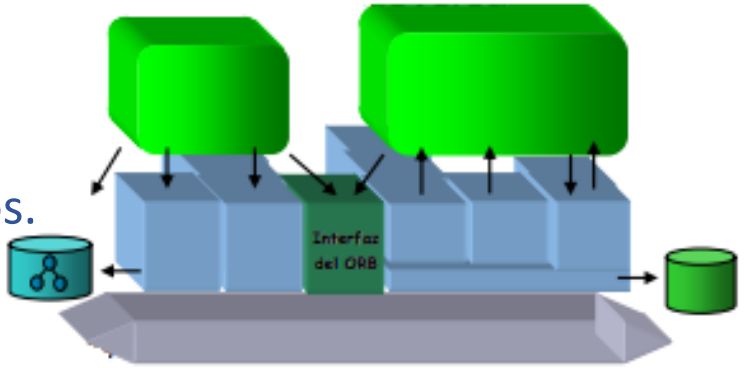


# Arquitectura de CORBA

## Lado cliente (3/5)

### Interfaz ORB

- Permite acceder a un conjunto de librerías o APIs que definen funciones del ORB y que pueden ser accedidas por el código cliente.
- Acceso a servicios iniciales como el servicio de nombrado
- Permite recuperar la información sobre una interfaz (metadatos)
- Permite descubrir en run-time los métodos que han de ser invocados.
- Conversión de referencias de objetos en cadenas y viceversa

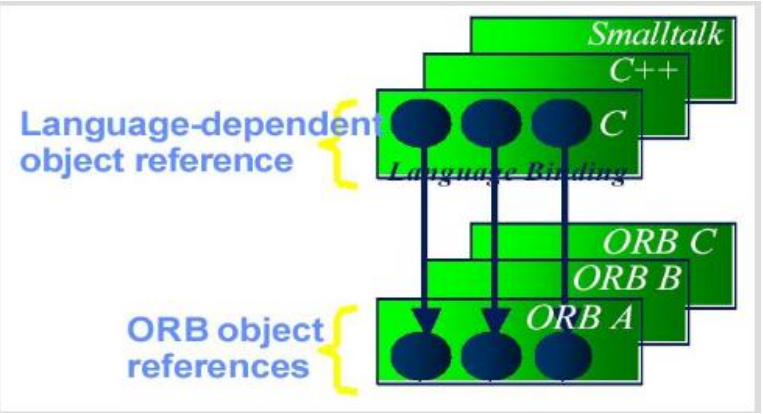


# Arquitectura de CORBA

## Lado cliente (4/5)

### Referencia de objetos

- Información necesaria para un cliente que desea interoperar con un target object a través del ORB.
- Target Object es el objeto al cual un cliente hace una petición, la implementación de dicho objeto no es definida por CORBA.
- Para evitar conflictos entre ORB's de diferentes vendedores CORBA 2.0 define los IOR's (Interoperable Object References).

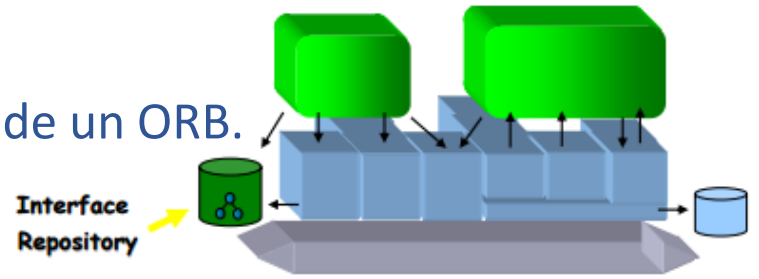


# Arquitectura de CORBA

## Lado cliente (5/5)

### Repositorio de interfaces

- Base de datos en tiempo de ejecución con versiones máquina de las interfaces IDL (y posiblemente información asociada).
- Permite obtener y modificar dinámicamente las descripciones (interfaces, métodos y parámetros) de todos los objetos registrados.
- Es un mecanismo de auto-descripción (metadatos) de los objetos
- El repositorio de interfaces en si es un objeto más, accesible a través de un ORB.



# Arquitectura de CORBA

## Lado del servidor(1/3)

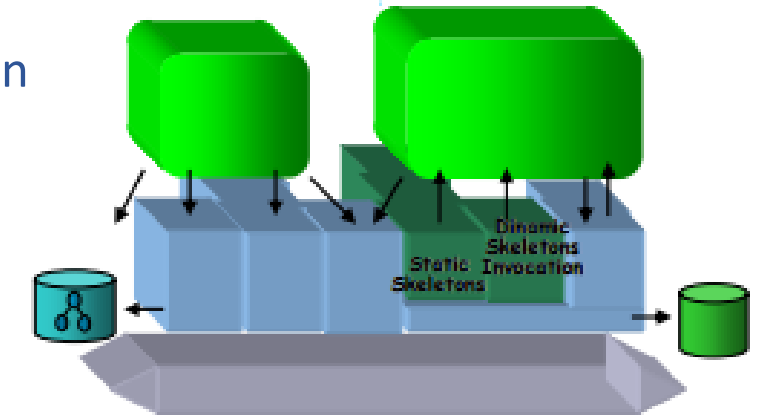
### Skeleton estático IDL

- Representante estático del cliente en el servidor.
- Generado en tiempo de compilación a partir de la interfaz IDL del servidor.
- Permite realizar el unmarshalling de las invocaciones del cliente.
- Para el servidor todas las llamadas son locales

### Interfaz de esqueletos dinámicos (DSI)

Da mecanismos de asociación en run-time a servidores que necesitan manejar peticiones de componentes que no tienen stubs

- Recibe las peticiones y averigua a que objetos van dirigidas.
- Son muy utilizados en la construcción de “Bridges” entre ORB’s.
- Se pueden usar para generar dinámicamente objetos.
- Es el equivalente en el servidor al DII del cliente.



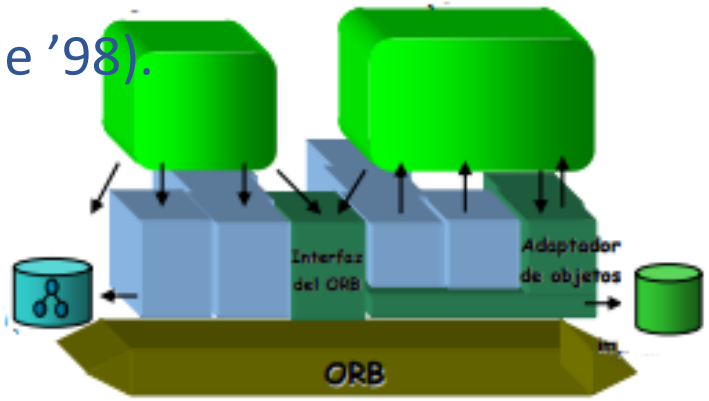
# Arquitectura de CORBA

## Lado del servidor(2/3)
## Adaptador de objetos

- Intermediario entre el núcleo del ORB y los objetos
- Instancia y activa objetos servidores, y crea referencias de objeto
- Mapea referencias de objetos a **IDs** de implementaciones de objetos
- Invoca el método apropiado cuando llega una petición remota (upcall).
- Registra objetos en el repositorio de implementaciones.
- POA (Portable Object Adapter): adaptador de objetos estándar (desde ’98).

## La interfaz ORB

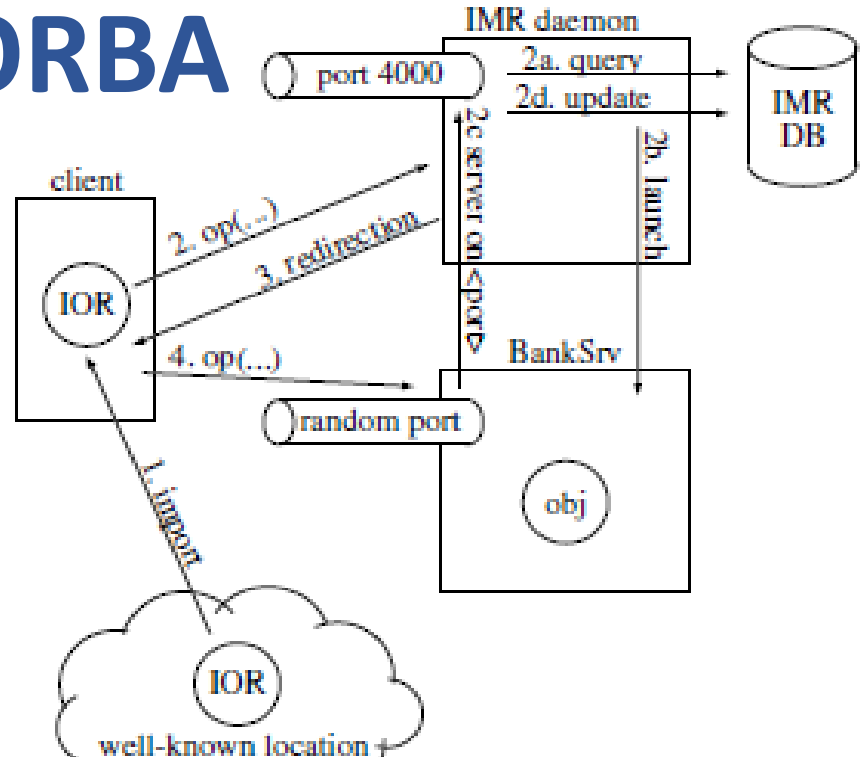
- API que da soporte local, idéntico al proporcionado en el lado del cliente, permite que el cliente y servidor interactúen con el ORB.

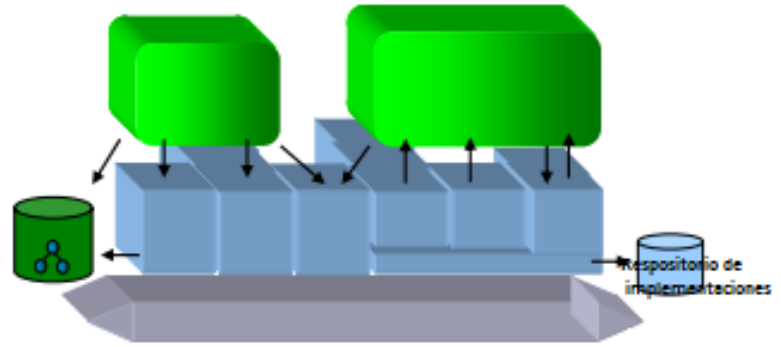


# Arquitectura de CORBA

## Lado del servidor(3/3)
## Repositorio de implementaciones

- Proporciona un registro en run-time de las clases, los objetos instanciados y los ID's que un servidor soporta.
- También puede almacenar otro tipo de información (trazas, datos administrativos)



# Arquitectura de CORBA

## Adaptadores de objetos estándarizados

### Basic Object Adapter (BOA, 1990)

Sub especificado ⇒ el API ofrecido a los sirvientes depende del vendedor del ORB

### Portable Object Adapter (POA, 1998)

Propósito principal: portabilidad

Código del servidor independiente de la implementación del ORB

# Arquitectura de CORBA

## Funcionalidades del POA

- El ORB entrega al POA una petición sobre un objeto CORBA
- Por cada servicio hay un POA, y de cada servicio puede haber varios objetos creados. cuando llega una petición para uno de los objetos de un servicio, es el POA de ese servicio sabe a cual de los objetos debe dirigirlo.
- Abre los mecanismos internos del servidor a los programadores
  - Da soporte flexible a la persistencia y activación de objetos
- Relaciona referencias de objeto, IDs de objeto y sirvientes
- Introduce la noción de Gestor de Sirvientes (Servant Manager)
  - Se ocupa de la creación, activación y desactivación de sirvientes

# Arquitectura de CORBA

## Activación con el POA

**Sirviente**

Una instancia de un objeto CORBA ejecutándose

**ID de objeto**

Referencia utilizada por el POA y la implementación de objetos para identificar a un objeto CORBA (y restaurar su estado cuando se activa)

**Referencia de objeto**

Referencia utilizado por un cliente CORBA que encapsula una dirección del nivel transporte, un ID del adaptador de objetos y un ID de objeto

**POA: relación flexible entre IDs de objeto y sirvientes**

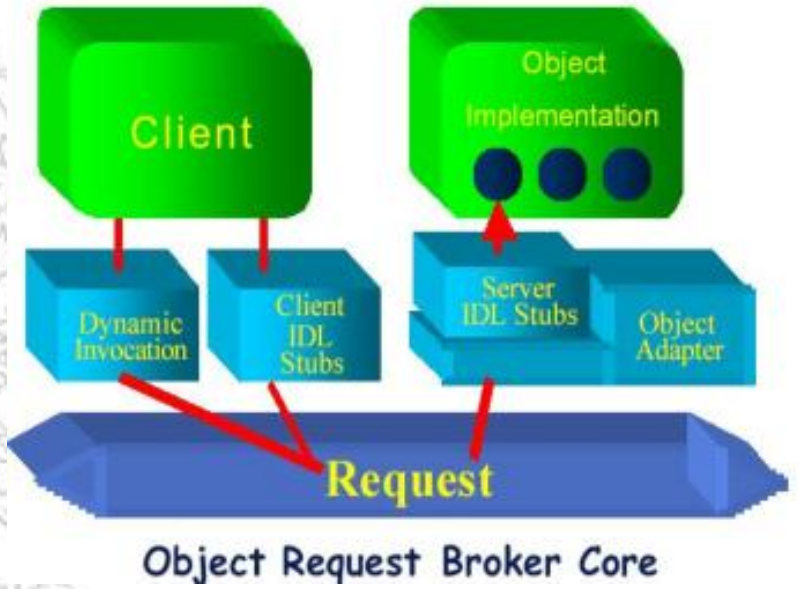
- Peticiones a distintos objetos son tratados por el mismo sirviente.
- Peticiones sucesivas al mismo objeto son tratados por distintos sirvientes

# Arquitectura de CORBA

## Invocaciones Estáticas vs Dinámicas

### Invocaciones Estáticas:

- Más fáciles de programar.
- Chequeo de tipos más robustos.
- Buen rendimiento.
- Auto documentado.




### Invocación Dinámica:

- Adición de clases al sistema sin hacer cambios en el cliente.
- Da soporte en la búsqueda de servicios en run-time.
- Permite escribir código genérico.

# Referencias


- Siegel Jon, OMG OVERVIEW: Corba and the OMA in Enterprise Computing.
- Venoski Steve, New Features for CORBA 3.0.
- Martinez, José Fernan, Introducción a CORBA, III Jornadas Ibero americanas en Telecomunicaciones y Telemática, 27-29 Agosto de 2001.
- Orfali, Robert; Harley, Dan; Client/Server Programming with Java and CORBA, Second Edition, WILEY Computer Publishing.
- Vinoski, Steve (1997). CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Communications Magazine. Febrero 1997.