Arquitecturas Distribuidas

TEMA 5. Otras arquitecturas distribuidas

II. Objetos distribuidos y CORBA

II. Objetos distribuidos y CORBA

1. Objetos Distribuidos
2. CORBA
    1. Características
    2. Modelo de trabajo
    3. ORB
    4. Arquitectura
    5. Localización de objetos
    6. Servicios de CORBA
    7. IDL

Objetos distribuidos

- Objeto: "entidad que encapsula información de estado privada o datos y un conjunto de operaciones asociadas para manipular los datos"
- Cada instancia de un objeto en memoria es una entidad diferenciada
  - Al ser una instancia tienen un ciclo de vida: se crean, usan y destruyen.
- Permiten descomponer las tareas en componentes más pequeños y reutilizables con un comportamiento bien definido.
- La aplicación de la POO en entornos distribuidos da lugar a la aparición de sistemas basados en objetos distribuidos.
  - Muy similar a las LPR, sólo que se invocan operaciones sobre objetos y normalmente se transfieren no sólo parámetros, sino objetos completos.
  - Los objetos siguen teniendo un **estado** y un **ciclo de vida**
- *DCOM, OLE, CORBA y Jini* son algunos ejemplos.

CORBA

- *Common Object Request Broker Architecture*
- Creado por el *Object Management Group* (OMG) en 1989
- Aboga por promover el uso de sistemas abiertos con interfaces estándar orientadas a objetos, sobre plataformas heterogéneas.
- Es una especificación de una sofisticada arquitectura de objetos distribuidos. Es independiente del lenguaje de programación.
- A partir de la especificación se desarrollan una serie de librerías de programación que proporcionan funcionalidad común (infraestructura).
- Existen implementaciones para distintos lenguajes de programación.

Características de CORBA

- Posee lenguaje específico para la definición de interfaces: IDL (*Interface Definition Language*)
  - Tipos: primitivos y estructurados
- El IDL es **independiente del lenguaje de programación**
- Un lenguaje de programación utiliza un **compilador de interfaces** específico para obtener los delegados del cliente y el servidor
- Forma de paso de parámetros:
  - Primitivos y estructurados: por valor (`in`) o copia/restauración (`out` e `inout`)
  - Objetos CORBA: se **identifican** por referencia ROR (Referencia a Objeto Remoto)

Características de CORBA

- Excepciones remotas (definidas en las interfaces)
- Semánticas:
  - **al-menos-una-vez** (por defecto)
  - **quizás** (opcionalmente) → `oneway` en IDL

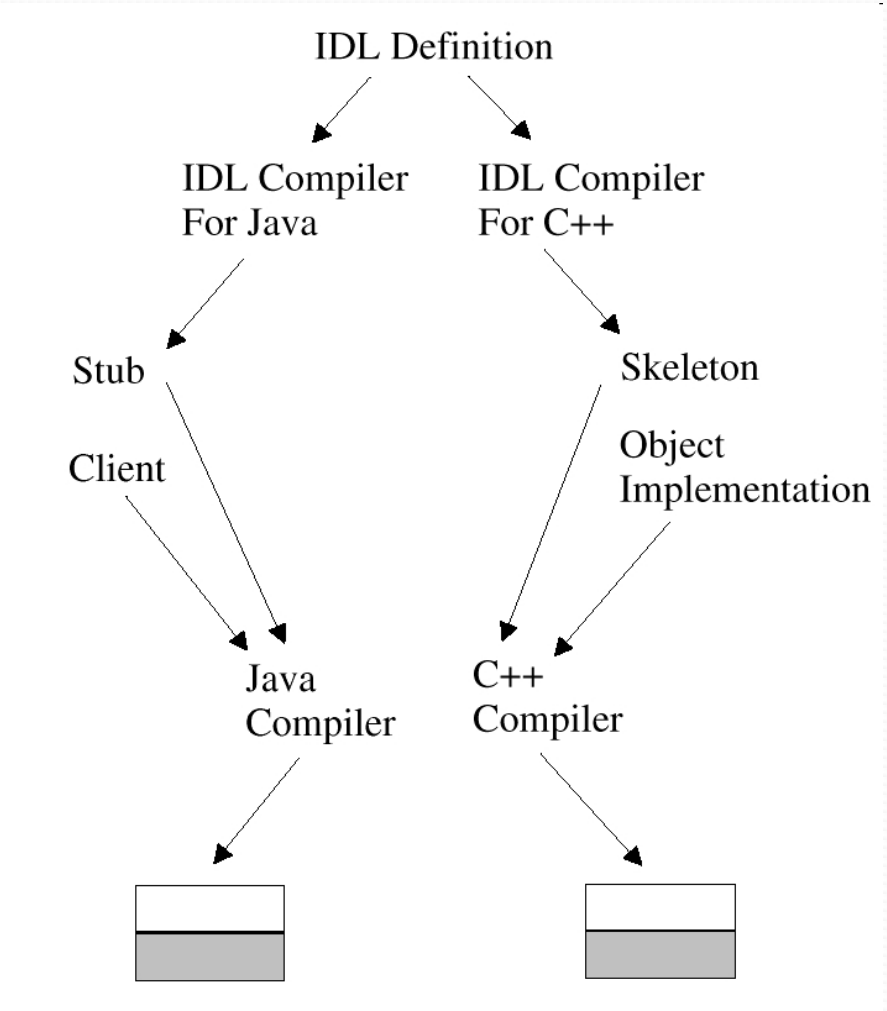
Características de CORBA

- Proporciona transparencia (para el programador):
  - De la plataforma
  - Del nivel de transporte
  - Del sistema operativo
  - Del lenguaje
  - De la implementación
  - De localización

Modelo de trabajo habitual

- El programador define las interfaces remotas, a través del lenguaje IDL (**independiente del lenguaje de programación**)
- El compilador de interfaces (CI) genera delegados para el cliente y el servidor (**en un lenguaje de programación en particular**)
  - Pueden usarse diferentes lenguajes en cliente y servidor
  - Ni el cliente ni el servidor tienen porque estar basados en un lenguaje OO

Ejemplo


IDL Definition
IDL Compiler For Java
IDL Compiler For C++
Stub
Skeleton
Client
Object Implementation
Java Compiler
C++ Compiler

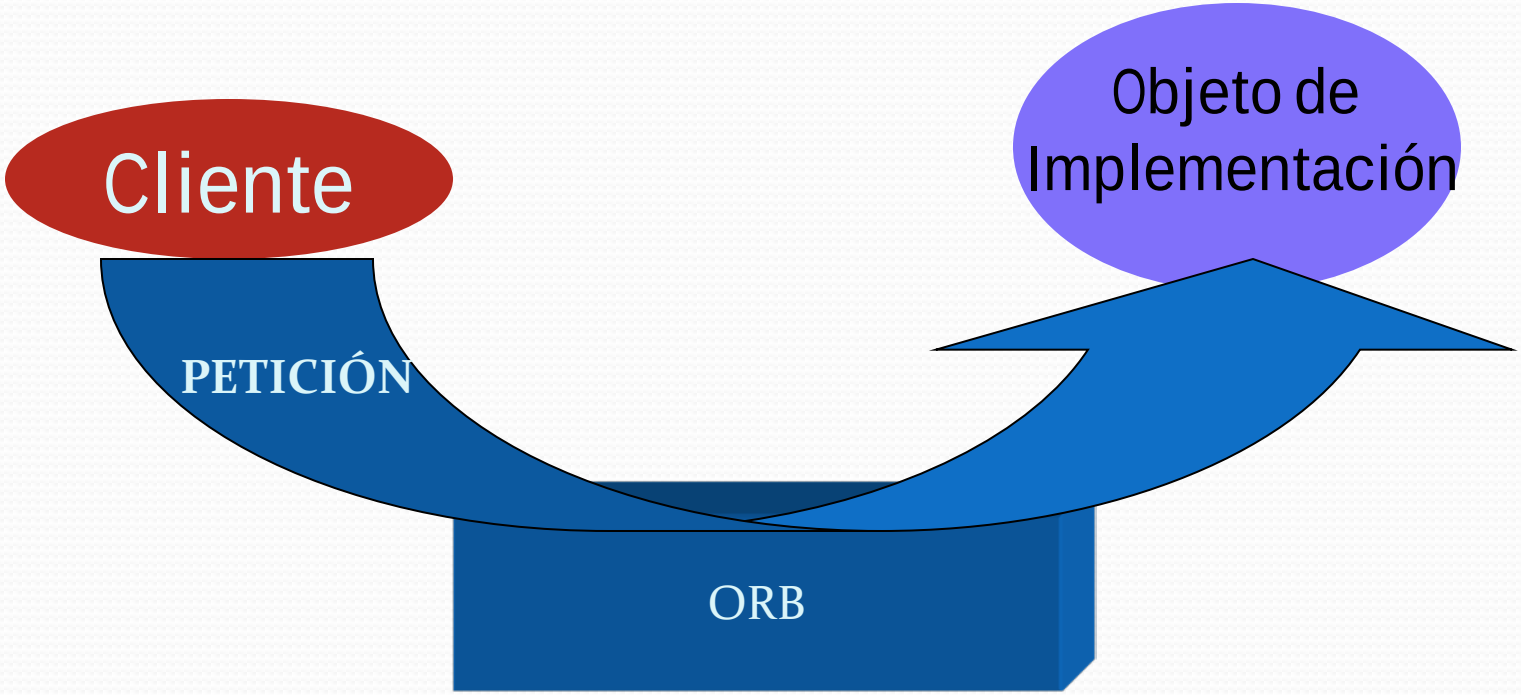
Modelo de trabajo habitual

1. El programador define las interfaces remotas, a través del lenguaje IDL
2. Servidor: se genera el delegado mediante el CI
3. Servidor: Se implementa el código del objeto servidor
4. Servidor: Se compila junto con el delegado creado, da lugar a un ejecutable
5. Cliente: se genera el delegado mediante el CI.
6. Cliente: Se implementa el código del objeto cliente
7. Cliente: se compila junto con el delegado creado, da lugar a un ejecutable

ORB

- La arquitectura de CORBA se basa en el uso de un componente que hace de puente entre diferentes plataformas llamado ORB
  - *Object Request Broker* (mediador en peticiones a objetos).
  - Encargado de la comunicación transparente entre clientes y los métodos de un objeto.
    - Localizar el objeto, activarlo y pasarle la solicitud.
  - Es un servicio (programa) que se ejecuta tanto en el cliente como en el servidor.

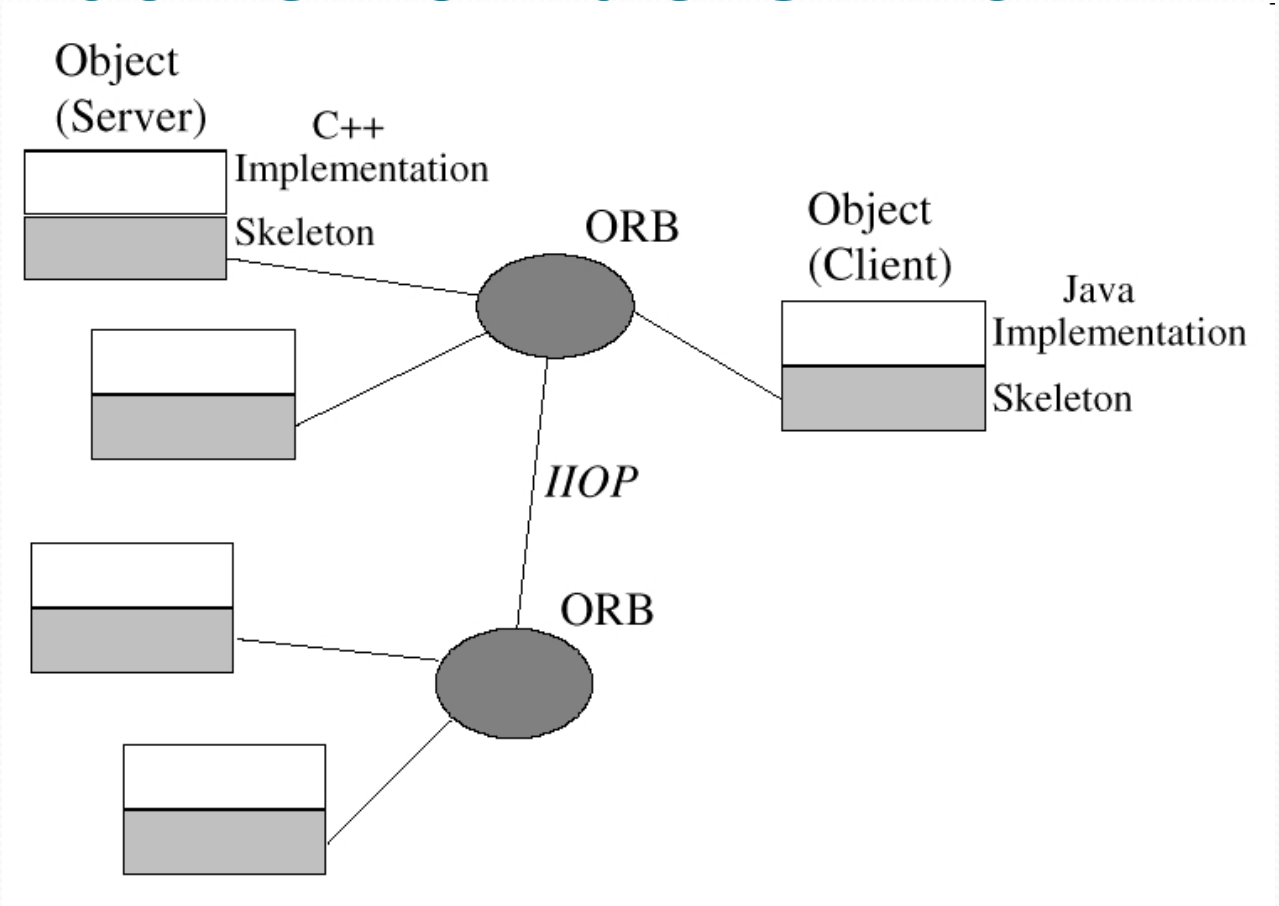
Arquitectura



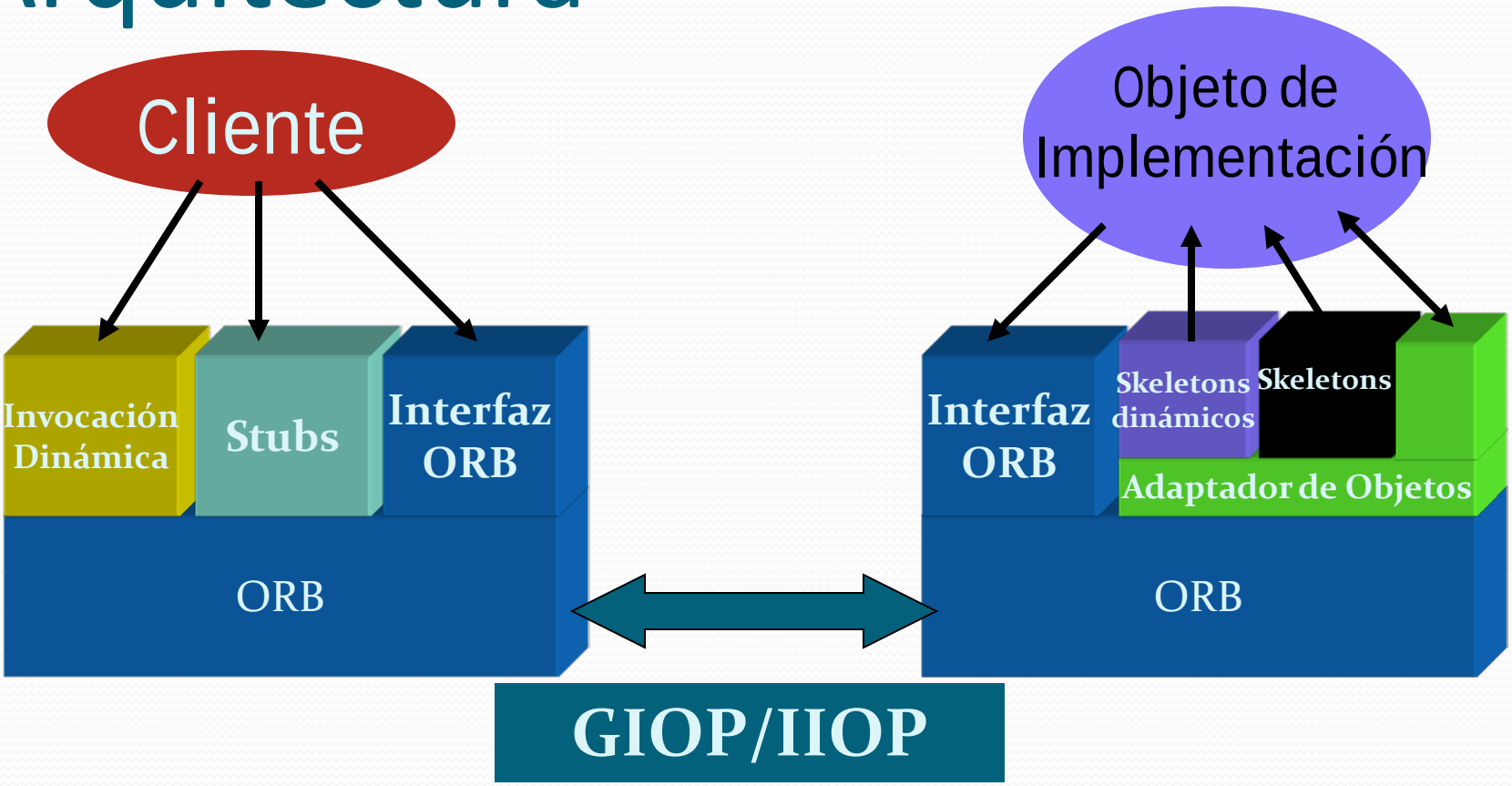
Arquitectura

- Versión inicial de CORBA aprobada en 1991
- En 1996: CORBA 2.0
  - GIOP (*General Inter-ORB Protocol)*
    - Normas que permiten que puedan cooperar implementaciones de diferentes fabricantes
    - Implementado sobre cualquier nivel de transporte
    - En Internet, sobre TCP/IP: IIOP (*Internet IOP*)

Interconexión de ORBs



Arquitectura



Localización de objetos

- ¿Cómo puede contactar un cliente con un objeto?
  - Necesita una referencia ROR a dicho objeto
  - Cada objeto posee ROR única
  - ROR es generada en el Adaptador de Objetos
  - Por medio de la ROR el ORB es capaz de localizar el ordenador remoto y el adaptador de objetos donde se encuentra. Este último es capaz de identificar el objeto concreto dentro del adaptador.

Localización de objetos

- En CORBA 2.0 se definen las ROR como:
  - Referencias a objetos interoperables (IOR):

NOMBRE DEL TIPO DE LA INTERFAZ IDL	PROTOCOLO DE TRANSPORTE Y DETALLES DE LA DIRECCIÓN			CLAVE DEL OBJETO	
IDENTIFICADOR DE DEPOSITO DE INTERFAZ	IIOP	NOMBRE DEL ORDENADOR	NÚMERO DEL PUERTO	NOMBRE DEL OA	NOMBRE DEL OBJETO EN EL OA

  - IOR intercambiado a través del ORB tiene formato binario.
  - Es posible conseguir un IOR "modo cadena" a partir del formato binario y vicecersa:

```
string_to_object, object_to_string
```

Localización de objetos

- ¿Cómo obtener ROR de un objeto?
  - A través de servicios especiales de CORBA
  - A partir de una IOR modo cadena:
    - Desde un archivo: por NFS
  - A través de un protocolo de descarga de objetos: HTTP, FTP, OBEX, etc.

Servicio de Nombres

- Permite:
  - Enlazar nombres a ROR (páginas blancas)
  - Creación de contextos de nombramiento (CN)
- Posee estructura jerárquica (como DNS)

Servicios CORBA

- Conjunto de utilidades genéricas para cualquier aplicación distribuida.
- Los más importantes son:
  - Nombres
  - Eventos
  - Notificación
  - Seguridad
  - Comercio
  - Transacciones
  - Control de la Concurrencia
  - Objetos persistentes

Interface Definition Language

- Permite definir
  - Módulos, interfaces, tipos, atributos y métodos.
- Subconjunto de C++ con construcciones adicionales
- Módulo
  - Permite agrupar interfaces y definiciones de tipos en unidades lógicas
  - Define un espacio de nombres

Interface Definition Language

- 15 tipos primitivos:
  - `octet, short, long, unsigned short, unsigned long, float, double, char, boolean`
  - Constantes de estos tipos: declaradas con `const`.
- 6 tipos estructurados
  - Pasados siempre por valor
  - Arrays y secuencias: siempre definidos con `typedef`
- any puede representar cualquier tipo primitivo o estructurado

Tipos estructurados en IDL

Tipo	Ejemplos	Uso
secuencia	`typedef sequence <Figura, 100> Todas;` `typedef sequence <Figura> Todas;`	Define un tipo para una sucesión de longitud variable de elementos de un tipo *IDL* especificado. Se puede especificar un límite superior de la longitud
string	`string nombre;` `typedef string<8> cadenaCorta;`	Define una cadena de caracteres terminada en nulo. Se puede especificar un límite superior de la longitud
array	`typedef octet idUnico[12];` `typedef ObjetoGrafico OG[10][8];`	Define un tipo para un vector de longitud fija de elementos de un tipo *IDL* especificado

registro	`struct ObjetoGrafico {`
	`  string tipo;`
	`  Rectangulo region;`
	`  boolean relleno;`
	`}`
		Define un tipo para un registro o estructura. Se pasan por valor en los argumentos y resultados

enumerado	`enum Rand {Exp, Numero, Nombre};`	El tipo enumerado en *IDL*

unión	`union Exp switch (Rand) {`
	`  case Exp: string voto;`
	`  case Numero: long n;`
	`  case Nombre: string s;`
	`}`
		La unión discriminada de *IDL* permite pasar como argumento a uno de entre un conjunto de tipos. La cabecera se parametriza con un *enum*, que especifica qué miembro se está usando.

Ejemplos IDL

```
//
// holamundo.idl
//

module hola {
        interface holamundo {
                void saludo();
        };
};
```

Ejemplos IDL

```
//
// holamundo.idl
//

module hola {
        interface holamundo {
                void saludo();
        };
};
```

Se traduce a un **package** Java

Ejemplos IDL

```
//
// holamundo.idl
//

module hola {
        interface holamundo {
                void saludo();
        };
};
```

Se traduce a un `interface` Java

Declaración de métodos en IDL

```
interface nombre {
      tipo metodo1(in/out/inout tipo par1,
                   in/out/inout tipo par2,
                   ...);
      tipo metodo2(in/out/inout tipo par1,
                   in/out/inout tipo par2,
                   ...);
      ...
      tipo metodoM(in/out/inout tipo par1,
                   in/out/inout tipo par2,
                   ...);
};
```

Referencias y bibliografía

- Libros:
  - "Client-server programming with Java and CORBA", Robert Orfali, Dan Harkey, 2nd ed, *John Wiley & Sons*, 1998.
  - "ORBacus: User Guide v. 4.2.1", *IONA Technologies PLC*. Disponible en:
    - http://labit501.upct.es/ad/manuales/CORBA/ob_421.pdf

Referencias y bibliografía

- IDL:
  - "AGETOR IDL guide", AGETOR. Disponible en:
    - http://ait.upct.es/ad/manuales/IDL/IDL_guide.htm