

CORBA

Davide Lamanna

davide.lamanna@isf-roma.org

MIDLAB

Roberto Baldoni

baldoni@dis.uniroma1.it

Agenda

- ❖ Introduzione architettura CORBA
 - ★ Elementi di base
 - ★ Interazione C/S
- ❖ Hello Server in CORBA
- ❖ Dettagli architettura CORBA
 - ★ Servizi CORBA
 - ★ Aspetti dinamici
 - ★ Interceptor

Architettura di base

- ❖ Middleware orientato agli oggetti
- ❖ Un oggetto CORBA è un'entità astratta che fornisce servizi
- ❖ CORBA introduce diversi livelli di *trasparenza*

- ★ *Trasparenza di linguaggio*

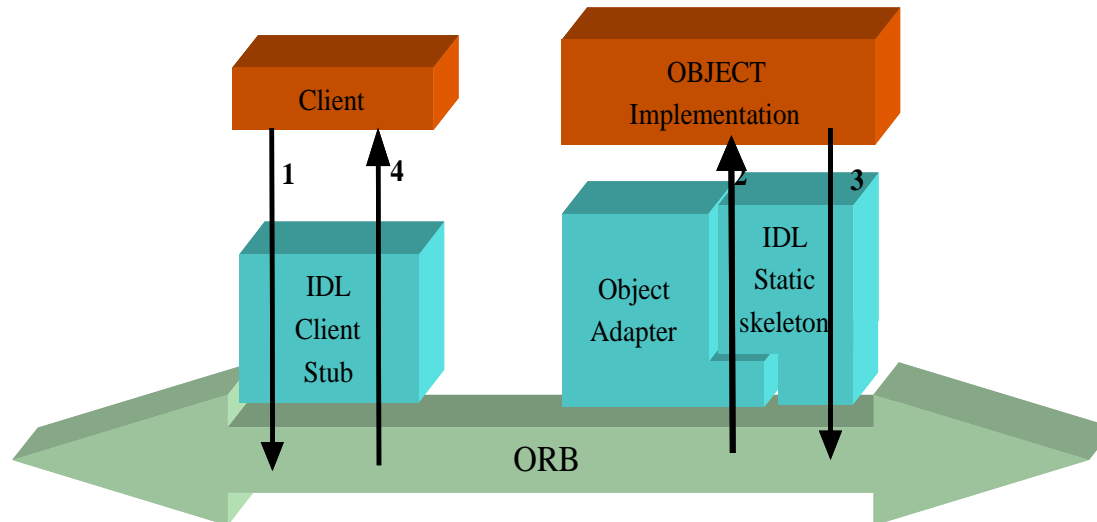
→ IDL

- ★ *Trasparenza di comunicazione*

→ ORB

- ★ *Trasparenza di locazione*

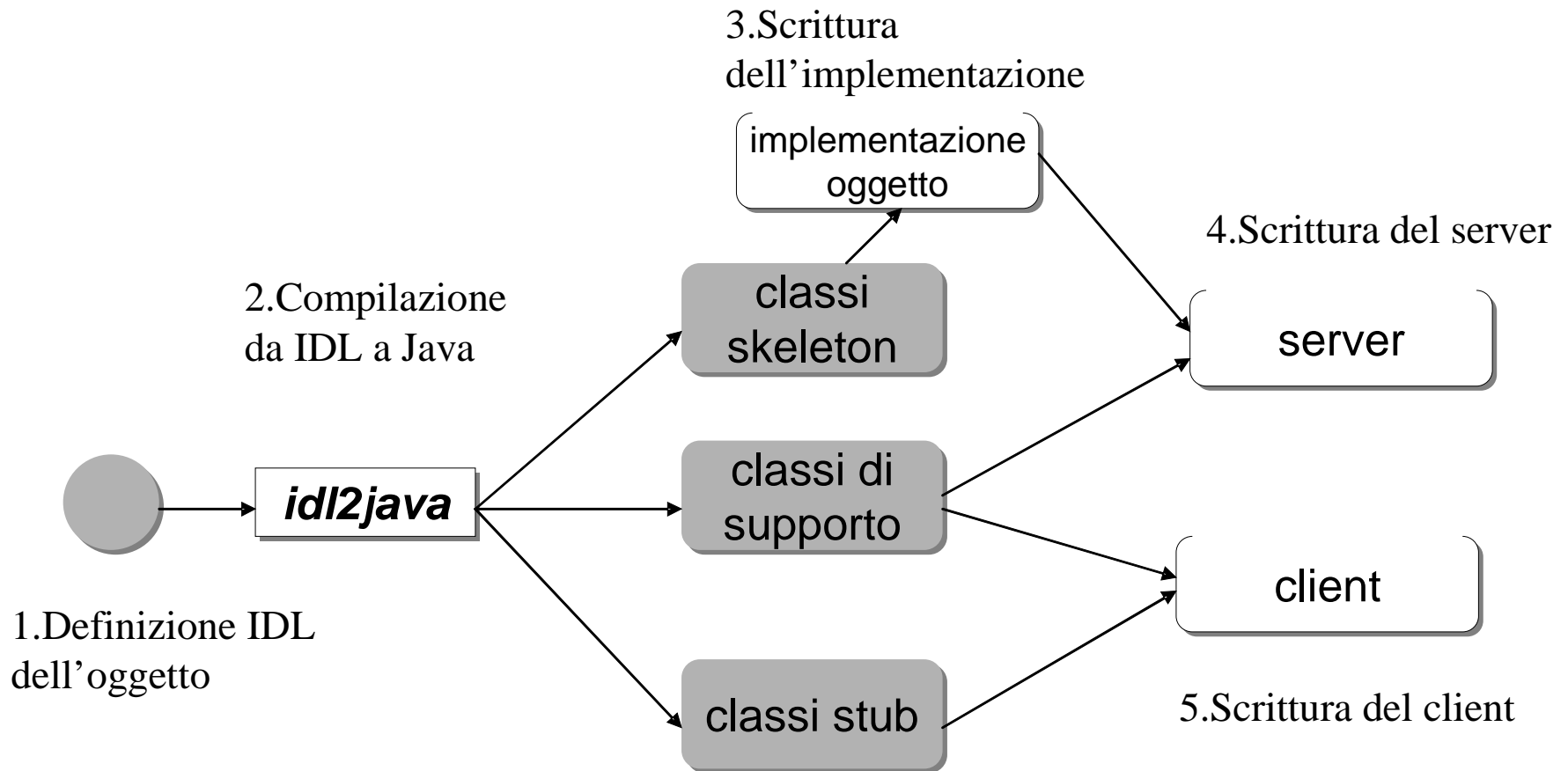
→ IOR



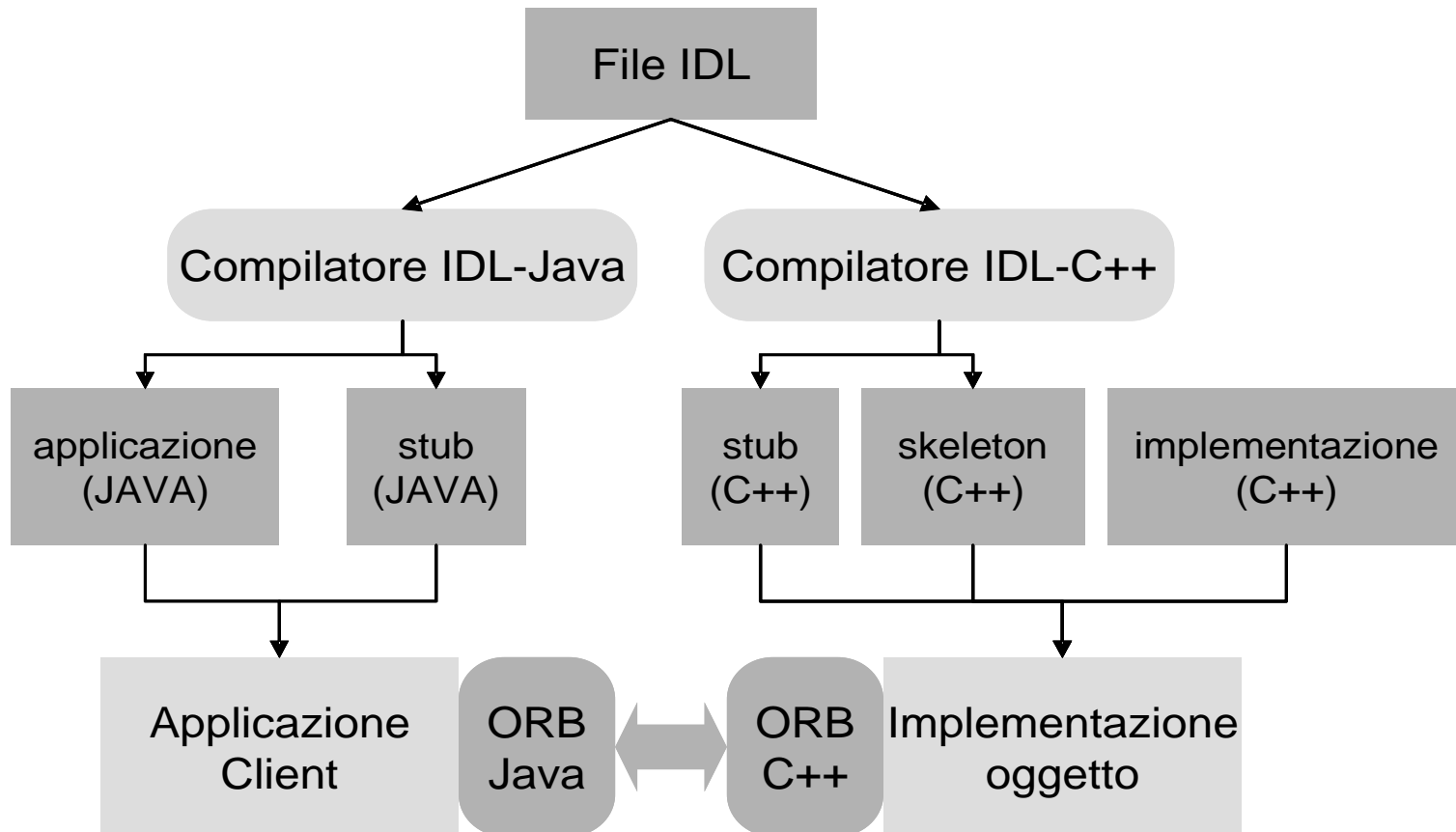
IDL – Interface Definition Language

- ❖ Un oggetto CORBA si manifesta attraverso la sua *interfaccia*
 - ★ IDL: linguaggio per definire l'interfaccia di un oggetto CORBA
- ❖ Tramite l'IDL si realizza la trasparenza di linguaggio

IDL – Sviluppo applicazione CORBA in Java

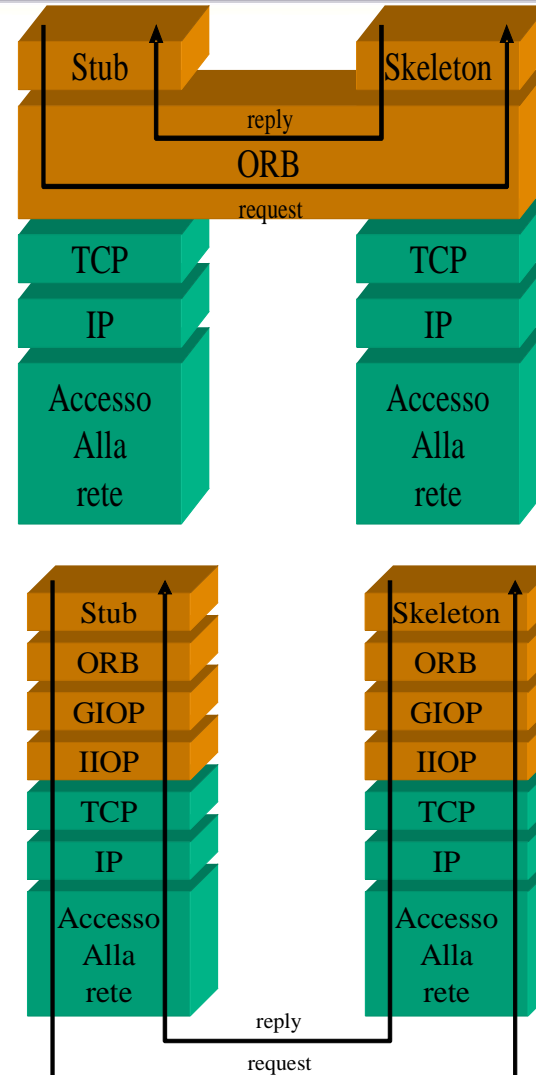


IDL – Sviluppo applicazione CORBA in Java/C++



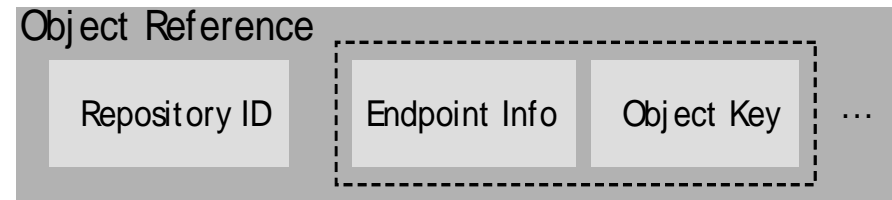
ORB – Object Request Broker

- ❖ L'ORB è "logicamente" un bus di comunicazione
- ❖ "Fisicamente" ogni processo CORBA ha un suo ORB
 - ★ Ogni invocazione CORBA passa per l'ORB
 - ★ È l'ORB che gestisce la comunicazione



IOR – Interoperable Object Reference

- ❖ Ad ogni oggetto server è associato un IOR.
- ❖ Lo IOR contiene informazioni di indirizzamento dell'oggetto.
- ❖ Per invocare operazioni su un oggetto (server), il client deve ottenere il suo IOR.
- ❖ Al momento dell'invocazione, l'ORB legge le informazioni nell'IOR e instaura la connessione con l'oggetto



- ❖ Repository ID: rappresenta il tipo IDL dell'oggetto
- ❖ Endpoint Info: la coppia ***host:porta***
- ❖ Object Key: identificativo dell'oggetto

POA: Portable Object Adapter

- ❖ Il POA è responsabile dell'associazione tra gli oggetti CORBA e le loro implementazioni in un linguaggio di programmazione (*servants*)
- ❖ In pratica, le funzioni del POA sono:
 - ★ creare gli Object Reference
 - ★ attivare gli oggetti
 - ★ inviare le richieste effettuate su un oggetto al rispettivo servant

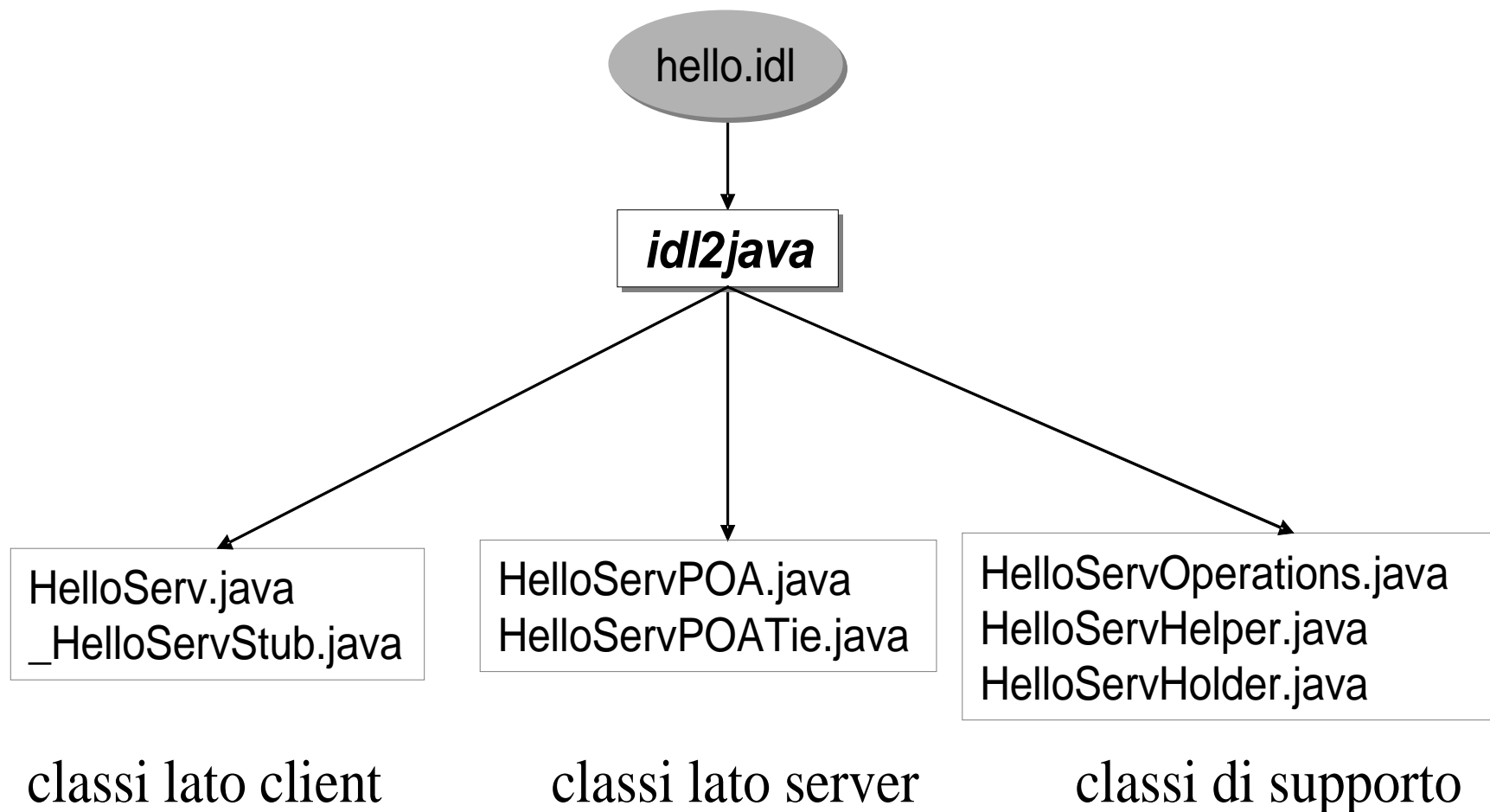
Un semplice esempio: Hello Server

1. Definizione IDL dell'interfaccia dell'oggetto

```
interface HelloServ {  
    string hello(in string param);  
};
```

Un semplice esempio: Hello Server

1. Compilazione idl 2 Java



Un semplice esempio: Hello Server

1. Implementazione dell'oggetto

```
public HelloImpl extends HelloServPOA
{
    public String hello(String param)
    {
        return "Hello: "+param;
    }
}
```

Un semplice esempio: Hello Server

1. Scrittura del server

```
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
public class Server
{
    public static void main(String[] args) throws Exception
    {
        ORB orb = ORB.init(args,null);
        POA rootPOA = POAHelper.narrow(
            orb.resolve_initial_references("RootPOA"));
        rootPOA.the_POAManager().activate();
        org.omg.CORBA.Object o = rootPOA.servant_to_reference(
            new HelloImpl());
        PrintWriter ps = new PrintWriter(
            new FileOutputStream(new File("hello.ior")));
        ps.println(orb.object_to_string(o));
        ps.close();
        orb.run();
    }
}
```

inizializzazione dell'ORB

inizializzazione del POA

Creazione dell'oggetto CORBA

scrittura in un file del riferimento dell'oggetto

attivazione dell'ORB

Un semplice esempio: Hello Server

1. Scrittura del client

```
import org.omg.CORBA.*;
public class Client
{
    public static void main(String[] args) throws Exception
    {
        HelloServ helloServ;
        BufferedReader br = new BufferedReader(new InputStreamReader(
            new FileInputStream("hello.ior")));
        String ior = br.readLine();
        br.close();
        ORB orb = ORB.init(args,null);
        org.omg.CORBA.Object o = orb.string_to_object(ior);
        helloServ = HelloServHelper.narrow(o);
        System.out.println(helloServ.hello("Ecco qua"));
        orb.shutdown(true);
        System.exit(0);
    }
}
```

Lettura dal file del riferimento dell'oggetto

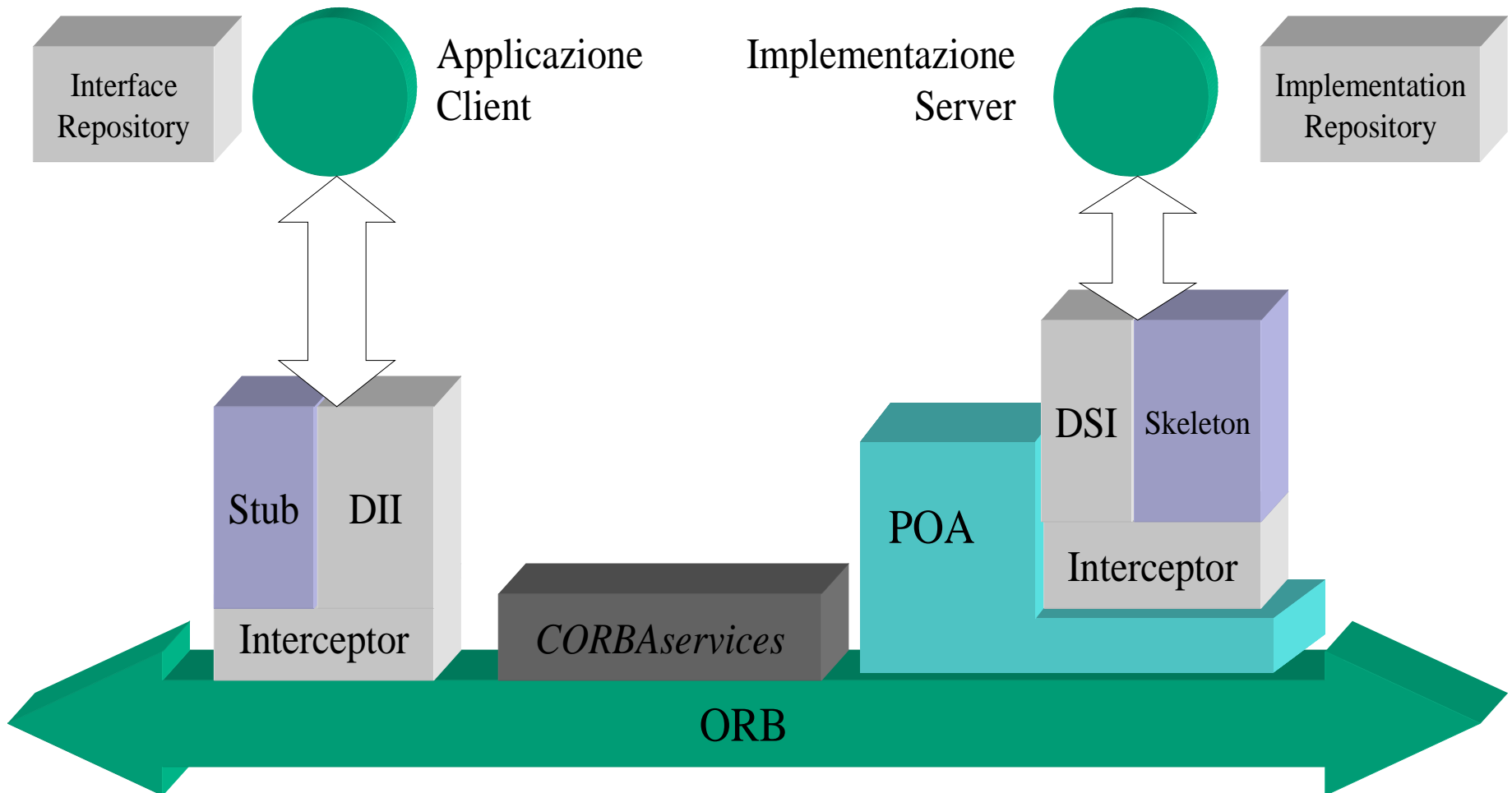
inizializzazione dell'ORB

Creazione riferimento dell'oggetto

Invocazione del metodo

Shutdown dell'ORB

Architettura CORBA completa



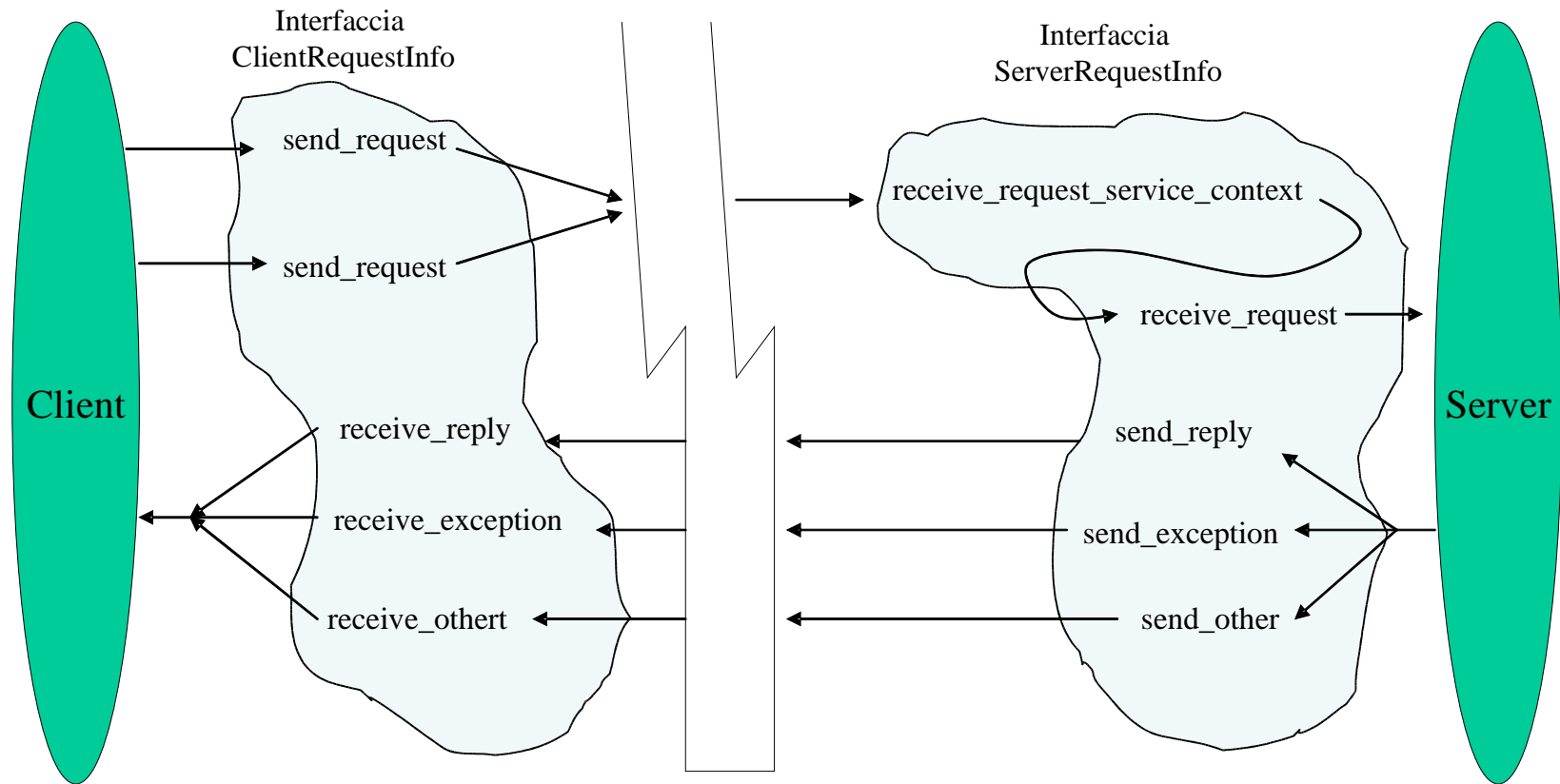
Portable Interceptor

- ❖ Oggetti che intervengono nel flusso della comunicazione tra client e server
- ❖ NON SONO oggetti CORBA
- ❖ “Catturano” richieste e risposte permettendo di leggerne i dati
- ❖ Diversi punti di intercettazione
 - ★ IORInterceptors
 - ★ *RequestInterceptors*

Request Interceptors

- ❖ Due tipi:
 - ★ Client-side
 - ★ Server-side
- ❖ Permettono di
 - ★ Leggere i dati di richieste/risposte
 - ★ Trasferire informazioni di contesto tra client e server
 - ★ Redirezionare le richieste a diversi server
- ❖ Ogni punto di intercettazione corrisponde ad un metodo dell'interceptor che viene automaticamente chiamato dall'ORB
- ❖ Il comportamento degli interceptor viene definito implementando gli opportuni metodi

Punti d'intercettazione

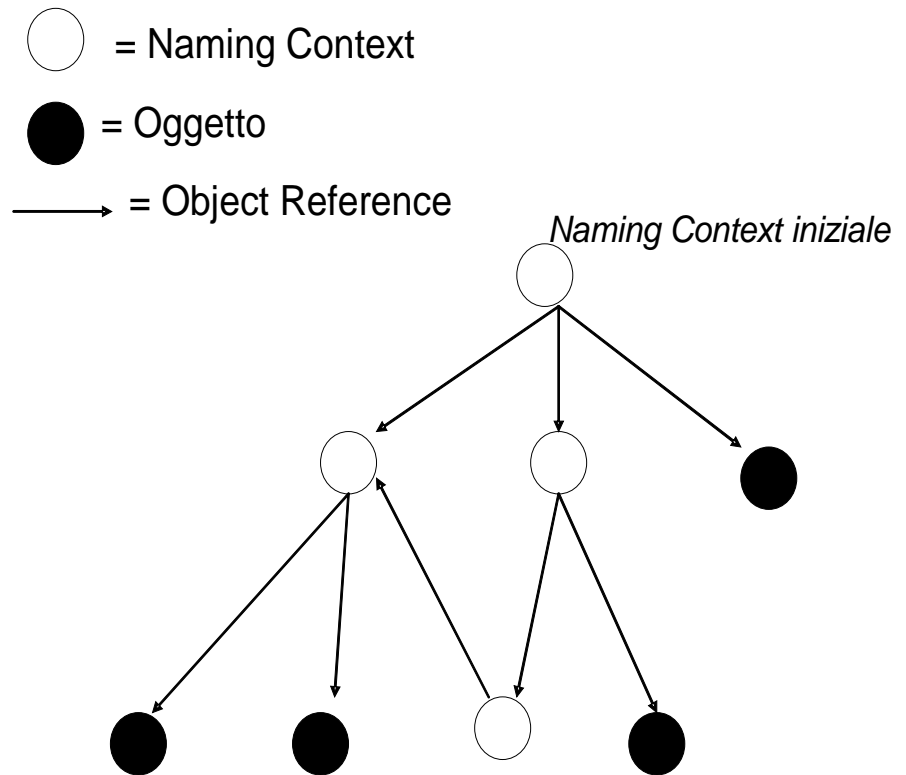


CORBA Services

- ❖ Servizi per sistemi distribuiti
 - ★ Event Service
 - ★ **Naming Service**
 - ★ Trading Service
- ❖ Servizi di supporto alle applicazioni
 - ★ Transaction Service
 - ★ Concurrency Service
 - ★ Persistence Service
- ❖ Servizi di tipo generale
 - ★ Time Service
 - ★ Life Cycle Service

Name Service

- ❖ Associa stringhe (nomi) ad object reference
 - ★ I client identificano i server semplicemente dal loro nome
 - ★ Indipendente dalla locazione del servizio
- ❖ Struttura a grafo



Name Service: l'interfaccia NamingContextExt

❖ Lato server

```
import org.omg.CosNaming.*;
```

```
NamingContextExt nc = NamingContextExtHelper.narrow(  
    orb.resolve_initial_references("NameService"));  
nc.bind(nc.to_name("HelloServer"), o);
```

❖ Lato client

```
import org.omg.CosNaming.*;
```

```
NamingContextExt nc = NamingContextExtHelper.narrow(  
    orb.resolve_initial_references("NameService"));  
HelloServ helloServ = HelloServHelper.narrow(  
    nc.resolve(nc.to_name("HelloServer")));
```

Gestione Naming Context annidati

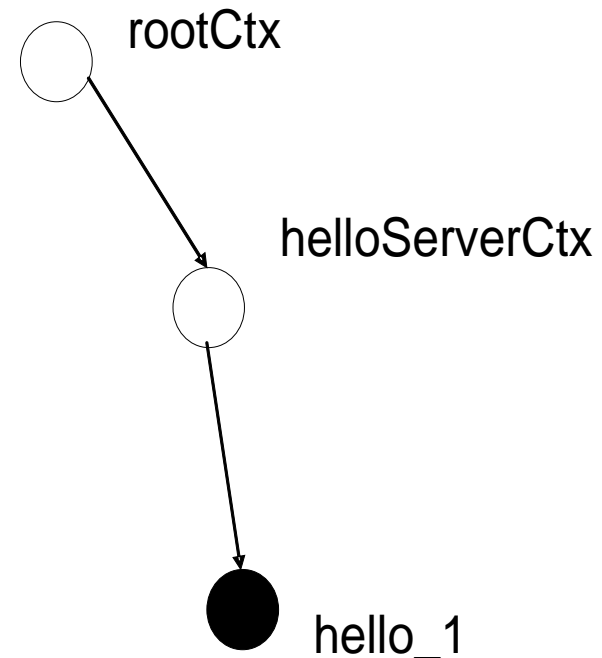
```
import org.omg.CosNaming.*;

org.omg.CORBA.Object o =
    poa.servant_to_reference(
        new HelloImpl());

NamingContextExt rootCtx =
    NamingContextExtHelper.narrow(
        orb.resolve_initial_references(
            "NameService"));

NamingContextExt helloServerCtx =
    rootCtx.bind_new_context(
        rootCtx.to_name("helloServerCtx"));

helloServerCtx.bind(
    helloServCtx.to_name("hello_1"), o);
```



L'interfaccia NamingContextExt

```
module CosNaming
{
    interface NamingContext
    {
        // ...
        void bind(in Name n, in Object obj)
            raises (NotFound, CannotProceed, InvalidName, AlreadyBound);

        Object resolve(in Name n) raises...

        void bind_context(in Name n, in Object obj) raises...
        NamingContext  bind_new_context(in Name n);
        NamingContext  new_context();

        void unbind(in name n) raises ...
        void rebind(in Name n, Object obj) raises ...
        void destroy();
    };
};
```