Développement d'applications réparties

Intergiciel CORBA

Khaled Barbaria
khaled.barbaria@ensta.org

Faculté des Sciences de Bizerte
GLSI3

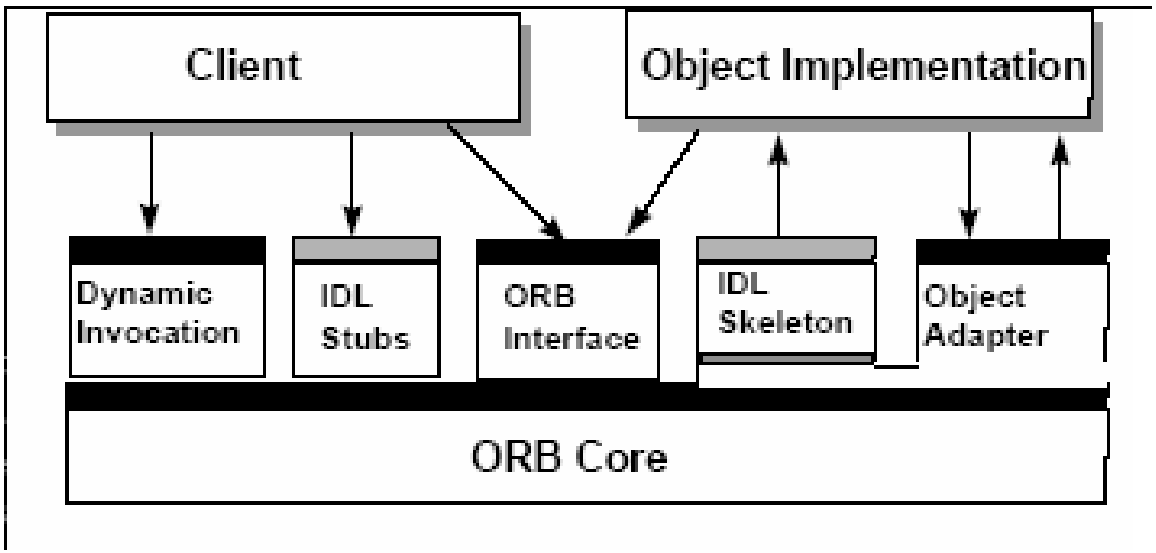
2021

Plan

Objets distribués: plan

Modèle des objets distribués (DOC)

- Les objets peuvent êtres locaux ou distants (espaces d'adressage différents)
- Les objets d'implémentation ont des interfaces visibles, et sont représentés coté client (patron proxy)
- Les objets distants sont masqués et sont vu comme locaux grâce aux souches et squelettes (générés à partir de l'interface du service)
- Invocation de méthodes distantes



Propriétés du modèle des objets distribués

- Support du modèle de programmation orienté objets
  - Objets, méthodes, interfaces, encapsulation, héritage, polymorphisme, exceptions, etc
- Modèle d'appel synchrone (comme pour RPC)
- Transparence de la localisation : Le middleware fait le mapping entre les références des objets et les localisations (possibilité d'utiliser un serveur de nommage)
- Services plus faciles à structurer et a exporter

CORBA: plan

2 CORBA

CORBA

CORBA : Common Object Request Broker Architecture

- Standard de l'OMG (Object Management Group)
- Indépendant des langages de programmation et des architectures matérielles

Architecture générale

- Object Request Broker (ORB) pour envoyer recevoir et exécuter les requêtes(coté client et coté serveur) .
- General Inter-ORB Protocol (GIOP) pour les comunications
- Interoperable Object References (IOR) pour référencer les objets (dans le contexte de l'application distribuée)
- CORBA Interface Definition Language (IDL)
- Stubs (proxies) and skeletons générés par un compilateur IDL.
- Dynamic remote method invocation

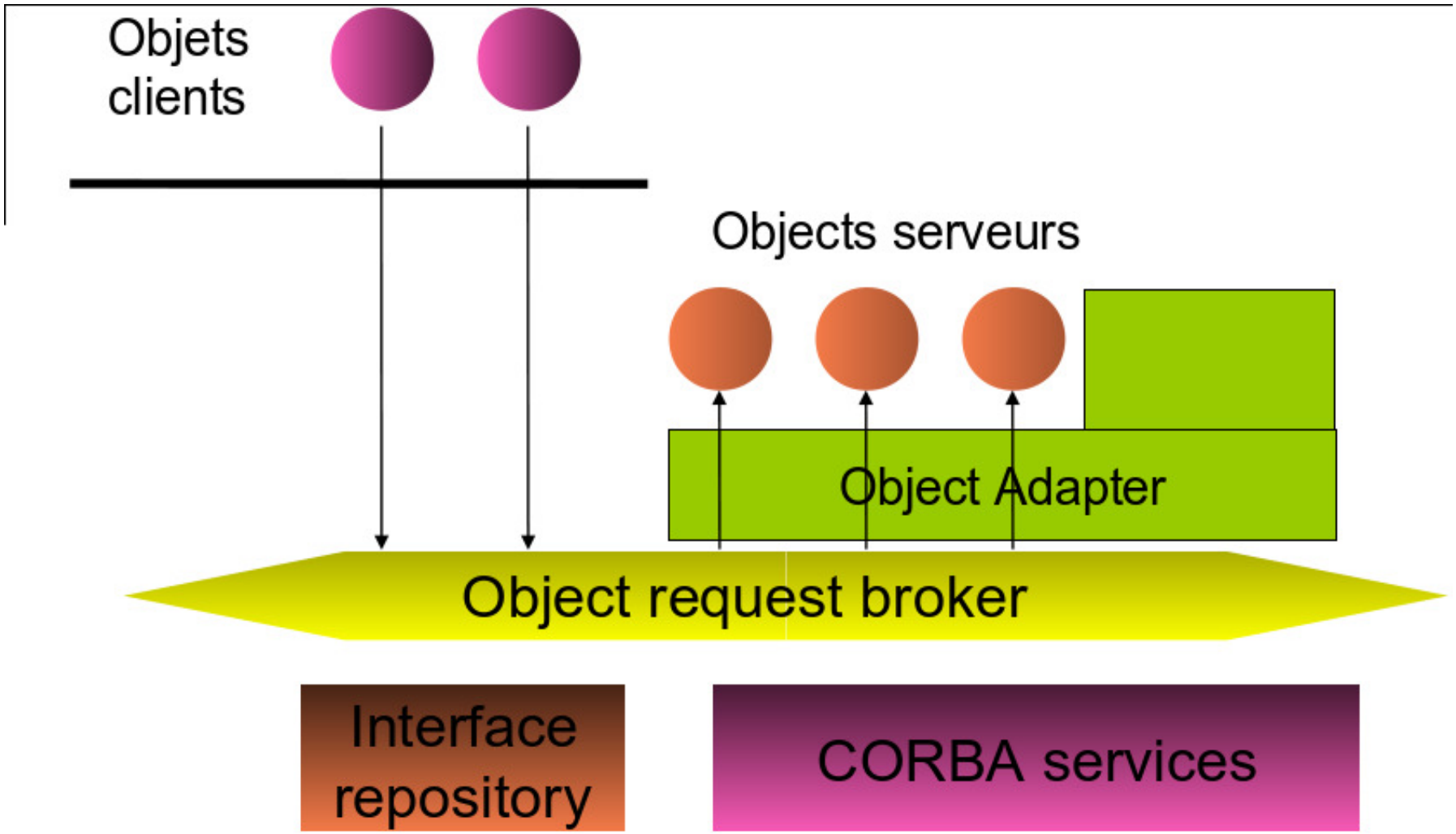
CORBA (2)

Autres services

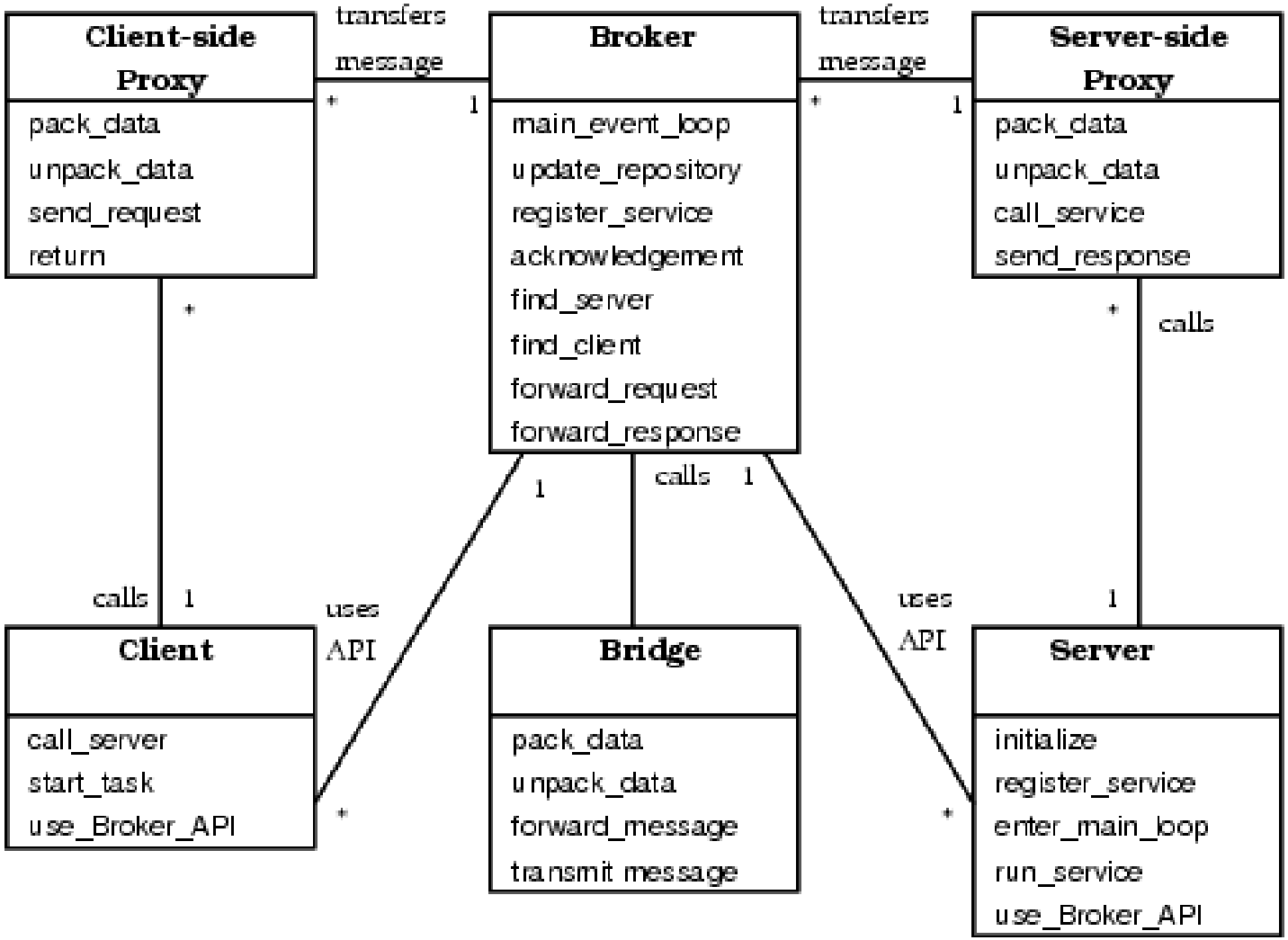
Dépôts d'interfaces et d'implémentations

- Interface Repository : retrouver (dynamiquement) les informations sur les types IDL (e.g. interfaces d'un objet) ce qui permet de construire les requêtes sans passer par les souches (DII, stub inutile dans ce cas)
- Implementation Repository : permet d'enregistrer les informations relatives aux services lancées sur les serveurs CORBA ainsi que la possibilité d'activer et de désactiver ces serveurs

Architecture CORBA



Patron de conception Broker



Interface Definition Language

Interface Definition Language

- Langage déclaratif neutre qui définit les interfaces (contrats) à réaliser par les objets
- la norme définit également des mapping standards vers C++, Java, Ada, Smalltalk, etc.
- Un compilateur génère les souches et squelettes en se basant sur le mapping IDL ,langage cible
- Il est possible d'utiliser plusieurs compilateurs au cas où l'interopérabilité entre langages est nécessaire

Interface Definition Language (2)

Système de typage

- Types de base : long (32 bit), long long (64 bit), short, float, char, boolean, octet, any, etc.
- Types de haut niveau : struct, union, sequence, array, Object

Modes de passage de paramètres

- in, out, inout
- Les différentes variables (type de base et type de haut niveau) sont passés par valeur
- Les objets sont généralement passés par référence

Services CORBA

- Service de nommage (Transforme un nom (IOR) en une référence d'objet)
- Service de persistance (Implémentation des objets CORBA persistants)
- Service de transactions (Les invocations sur les objets font partie de transactions, garantie des propriétés ACID)
- Service d'évènements et de notification
  - Réponses aux besoins pour les communications asynchrones (modèles en push/pull, notification selon critères, etc.)
  - Mais : construit au dessus de modèle de communication synchrone

Mapping IDL- java

IDL Type	Java Type
module	package
boolean	boolean
char, wchar	char
octet	byte
string, wstring	String
short,	short
long,	int
float	float
double	double
fixed	BigDecimal
enum, struct, union	class
sequence, array	array
interface (non-abstract)	signature interface and an operations interface, helper class, holder class
interface (abstract)	signature interface, helper class, holder class
exception	class
Any	org.omg.CORBA.Any
typedef	helper classes
readonly attribute	accessor method
readwrite attribute	accessor and modifer methods
operation	method

CORBA en pratique

Étapes de mise en place d'une application CORBA

- Définir l'interface IDL
- Compiler l'IDL pour générer souches et squelettes
- Implémenter l'interface
- Développer le serveur
- Développer le client
- Lancer le serveur et le client (et potentiellement le service de nommage)

Exemple d'interface et d'implémentation

Listing 1 – Exemple d'interface IDL

```
1 module HelloApp{
2     interface Hello{
3   string echo (in string s);};};
```

Listing 2 – Exemple d'implémentation de l'interface

```
1 package HelloApp;
2 import org.omg.CORBA.*;
3
4 public class HelloAppImpl extends HelloPOA{
5     public String echo(java.lang.String S) {
6   System.out.println("echo " +S);
7   return ("JacORB : "+ S);}
8 }
```

Exemple de serveur CORBA

Listing 3 – Serveur JacORB

```
1  package HelloApp;
2  import org.omg.CORBA.*;import org.omg.PortableServer.*;
3  import java.util.*;import java.io.*;
4
5  public class Serveur{
6      public static void main(String args[]) throws Exception{
7          ORB orb = ORB.init (args, null);
8
9           org.omg.CORBA.Object objPoa
10        = orb.resolve_initial_references ("RootPOA");
11
12    POA rootPOA = org.omg.PortableServer.POAHelper.narrow (objPoa);
13    rootPOA.the_POAManager().activate();
14
15    HelloAppImpl hello = new HelloAppImpl ();
16    org.omg.CORBA.Object obj = hello._this (orb);
17    System.out.println (orb.object_to_string(obj));
18    orb.run ();
19      }
20 }
```

Exemple de client CORBA

Listing 4 – Client JacORB

```
1  package  HelloApp;
2  import org.omg.CORBA.*;
3
4  public class Client{
5
6      public static void main(String args[]) throws Exception {
7
8    ORB orb = org.omg.CORBA.ORB.init(args,null);
9
10    HelloApp.Hello server =
11        HelloApp.HelloHelper.narrow(orb.string_to_object(args[0]));
12
13    System.out.println(server.echo("hello from jacorb client"));
14      }
15 }
```

Discussion

Avantages de CORBA

- Transparence de localisation et d'accès : utilisation des objets indépendamment des localisations
- Séparation entre interfaces et implémentations : clients dépendants des interfaces, pas des implantations
- Interfaces typées, Références d'objets typées par les interfaces
- Tous les avantages de l'OO ; même si les objets n'appartiennent plus à la même machine, et même si les architectures matérielles, les langages de programmation et les OS sont différents

Discussion(2)

Inconvénients de CORBA

- Couplage fort. Modèle d'interaction synchrone (requête/réponse) : Même si les sémantiques oneway et l'Asynchronous Method Invocation (AMI) sont prévus.
- Libération des objets distribués non utilisés ? Ramasse miettes distribués ?
- Modèle plutôt statique et lourd plusieurs services, parfois pas nécessaires (applications mobiles et embarquées)
- Modèle pouvant causer des problèmes de performance et de déterminisme
- Paradoxe de l'intergiciel : si les intergiciels permettent de résoudre les problèmes d'interoprabilité, ils sont sources d'une nouvelle source d'hétérogénéité (l'application devient dépendante du MW)