



Middleware de communication entre objets distribués

Conception d'Applications Hétérogènes Distribuées

Lionel Médini

Septembre-novembre 2015

Plan du cours

- Outils de programmation avancés
- Systèmes d'information distribués
 - Appel de méthodes distantes (RPC)
 - Objets hétérogènes distribués (CORBA)
 - Objets Java distribués (RMI, RMI/IIOP)
 - Synthèse
 - Frameworks Java (Struts, Spring, JEE 5)
- Objets transactionnels distribués
 - Exemples d'EJB 2 et de descripteurs de déploiement
 - EJB 3 : POJO et annotations

Objets distribués

Plan

> Généralités

Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

Approche orientée objet
Architectures distribuées } Objets distribués

- Exemples d'application : agence de voyage en ligne
- Exemples d'objets distribués
 - Logique applicative client (connexion, recherche commandes)
 - Gestion sécurisée des paiements
 - Gestion des réservations
 - Interrogation des fournisseurs de voyages
 - ...

Infrastructures *middleware*

Plan

- > Généralités
- Objets distribués hétérogènes
- Objets distribués homogènes
- Synthèse objets distribués
- Conclusion

- But : gestion des communications entre les objets hétérogènes *via* le réseau dans les architectures distribuées
- Exemples
 - RPC (Sun, Microsoft...)
 - CORBA (OMG)
 - RMI (Java)
 - RMI/IIOP (Java)
 - Java EE (Java)
 - .Net (Microsoft)

RPC

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Origine

- Créé pour le système de fichiers NFS
- Version « originale » : Sun RPC (libre)
- Dernière RFC : 1057, juin 1988

- Principe

- Appel de fonctions distantes
- Fonctionne sur un mode client-serveur
- Langages de programmation hétérogènes
- Transparence des appels distants pour les composants locaux

RPC

Plan

Généralités

> Objets distribués hétérogènes

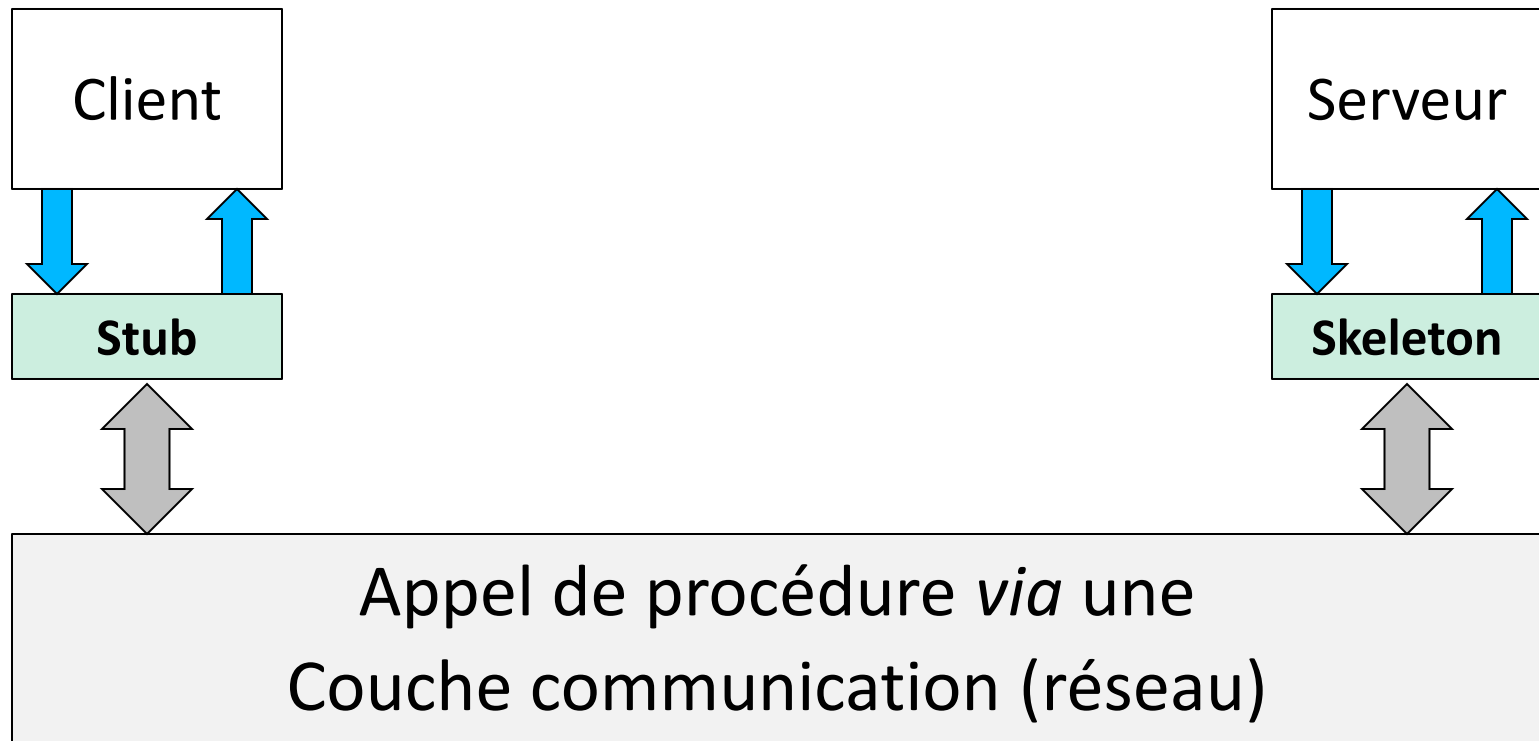
Objets distribués homogènes

Synthèse objets distribués

Conclusion

• Fonctionnement

- Des composants locaux (stub et skeleton) masquent la couche réseau au client et au serveur



RPC

Plan

Généralités

> Objets distribués hétérogènes

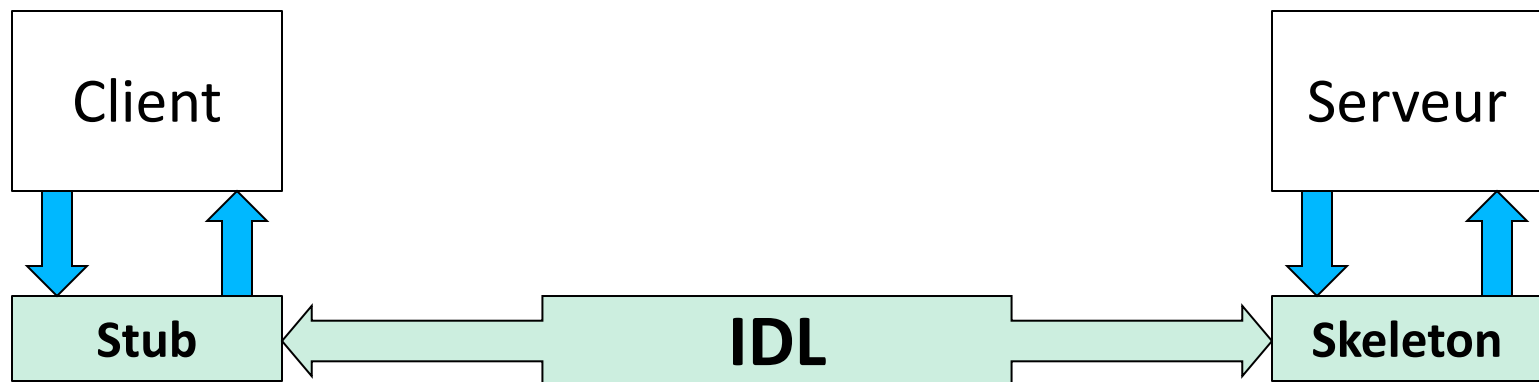
Objets distribués homogènes

Synthèse objets distribués

Conclusion

- **Fonctionnement**

- Au niveau logique, les prototypes des fonctions et types de données exposés sont décrits dans un langage commun (IDL)



- Langages/notions de base

- Stub/skeleton

- Proxys de gestion des communications entre objets
 - Stub : proxy côté client
 - reçoit et achemine les requêtes du client
 - récupère les réponses et les transmet au client
 - Skeleton : proxy côté serveur
 - Récupère les requêtes et les transmet au serveur
 - reçoit et achemine les réponses du serveur
 - Spécifiques aux langages, couches de communication, types d'objets

RPC

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Langages/notions de base
 - IDL : langage neutre de spécification d'interfaces
 - Langage commun de représentation des interfaces
 - Typage des données en XDR (eXternal Data Representation : dernière RFC : 4506)
 - Traduction (projection) des interfaces dans différents langages
 - C, C++, SmallTalk, Cobol OO, Java, Common Lisp...
 - Compilateur spécifique à chaque langage

RPC

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Conclusion
 - Limité à l'appel de méthodes (pas d'objet)
 - Introduit les principes fondamentaux
 - d'indirection dans l'accès aux ressources distantes
 - de **serveur d'application**
- Remarques
 - Plusieurs versions, incompatibles entre elles
 - Implémentation sur HTTP : XML-RPC
- Références
 - <http://www.ietf.org/rfc/rfc1057.txt>
 - <http://www.crevola.org/francois/?content=articles&show=1>

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- But
 - Communication entre objets hétérogènes et distants
 - Invocation de « services » entre objets d'applications distribuées
- Principes généraux
 - Séparation stricte Implémentation/Interface
 - Transparence par rapport à
 - À la localisation des objets
 - Au langage d'implémentation
 - Architecture client/serveur

CORBA

Plan

Généralités

- > Objets distribués hétérogènes
- Objets distribués homogènes
- Synthèse objets distribués
- Conclusion

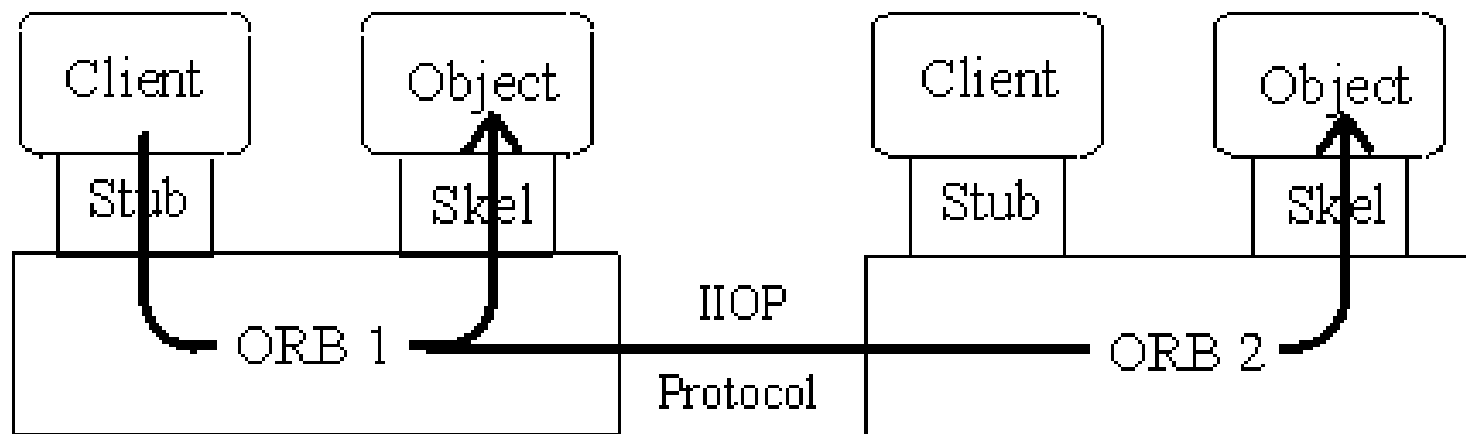


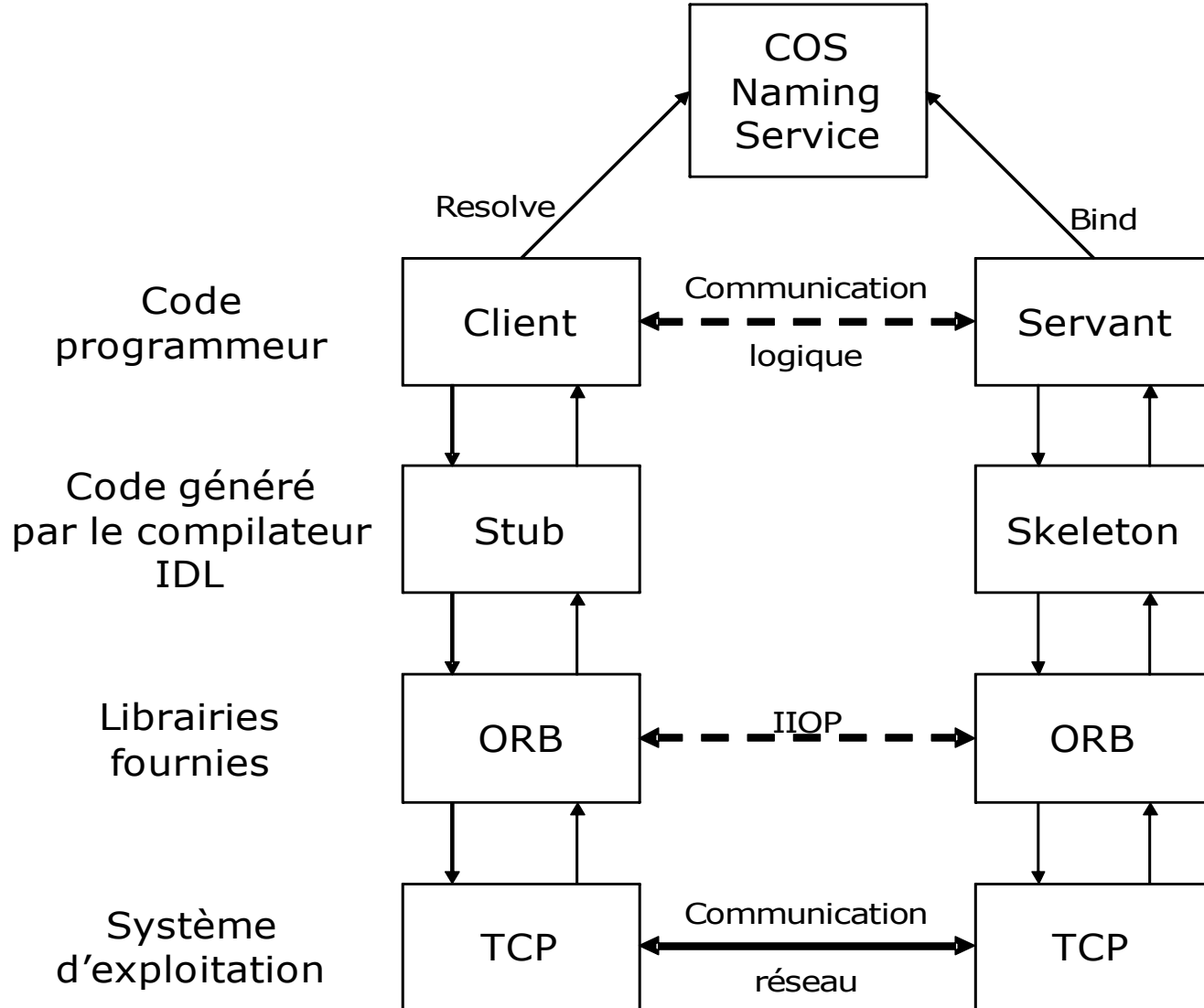
Figure 2: Interoperability uses ORB-to-ORB communication

CORBA

Plan

Généralités

- > Objets distribués hétérogènes
- Objets distribués homogènes
- Synthèse objets distribués
- Conclusion



CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Objets/librairies

- ORB : Couche « communication » intégrée aux objets

- Responsable des mécanismes nécessaires pour
 - Trouver l'implémentation de l'objet pour la requête
 - Préparer cette implémentation à recevoir la requête
 - Communiquer les données constituant la requête
 - Un ORB contient
 - Une interface IDL
 - Un support au service de nommage COS
 - Un support IIOP

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Objets/librairies
 - Stub : proxy client
 - Skeleton : proxy serveur s'interfacent avec un ORB
- CORBA COS : services objets communs
 - Naming Service : service de nommage permettant de retrouver les objets servants pour les clients
 - Life Cycle Service
 - Object Transaction Service
 - Security Service
 - ...

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Langages/protocoles
 - IDL (suite) : langage neutre de spécification d'interfaces
 - Décrit l'interface d'un servant
 - Génération
 - Effectuée à la compilation (statique)
 - À l'aide d'un compilateur IDL spécifique à chaque langage

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Langages/protocoles
 - IIOP : protocole de communication entre ORB
 - Transmission des messages
 - Implémentation de GIOP sur TCP
 - Échange de messages entre « ORB »

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Langages/protocoles
 - DII : accès dynamique aux serveurs
 - Permet de « court-circuiter » un ORB
 - Découverte dynamique de nouveaux objets
 - Construction et distribution d'invocation
 - Réception de réponses

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- Fichier Chat.idl

```
interface Chat {  
    void setMessage (in string auteur, in string texte);  
}
```

- Compilation

```
idlj -fallTIE Chat.idl
```

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation (Java)

- Interface que doit implémenter le servant

```
public interface ChatOperations {  
    void setMessage (String auteur, String texte);  
}
```

- Interface distante que doit implémenter le servant

```
public interface Chat  
    extends ChatOperations,  
        org.omg.CORBA.Object,  
        org.omg.CORBA.portable.IDLEntity  
{ ... }
```

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation (Java)
 - Éléments générés par le compilateur
 - Classe skeleton
`Chat_Tie`
 - Classe stub
`ChatStub`
 - Classe contenant des méthodes auxiliaires :
`ChatHelper`

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- À Programmer

```
class ChatServant implements ChatOperations {  
    void setMessages (String auteur, String texte) {...}  
}
```

```
class ChatServer {  
    public static void main (String args[]) {...}  
}
```

```
class ChatClient {  
    public static void main (String args[]) {...}  
}
```

CORBA

Plan

Généralités

> Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- Lancement

- Serveur de noms (machine m1)

- `tnameserv` (depuis JDK 1.3)

- Serveur (machine m2)

- `java ChatServer -ORBInitialHost m1`

- Client(machine m3)

- `java ChatClient -ORBInitialHost m1`

RMI

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Mêmes principes de base que CORBA
- Limité à Java (objets non hétérogènes)
- Plus de langage de spécification d'interfaces
- Protocole de communication : JRMP

RMI

Plan

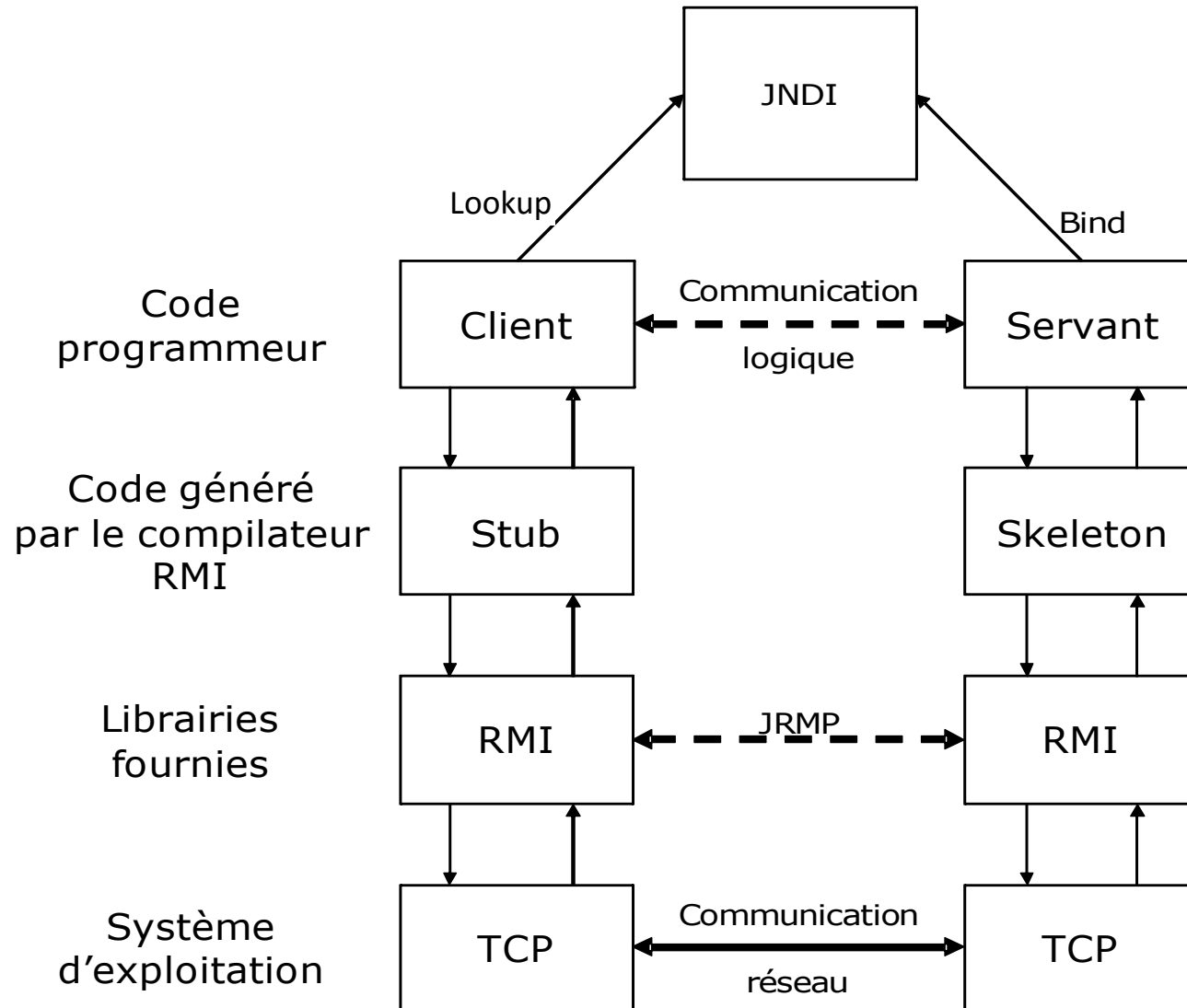
Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion



RMI

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- Interface Java

```
package monchat;  
import java.rmi.*;  
public interface Chat extends Remote {  
    public void setMessage (String auteur, String texte)  
    throws RemoteException;}
```

- Classe skeleton (supprimée depuis Java 1.5)

```
ChatServant_Skel
```

- Classe stub

```
ChatServant_Stub
```

RMI

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- À Programmer

```
class ChatServant extends UnicastRemoteObject implements
    Chat {
    void setMessages (String auteur, String texte){...} }
class ChatServer {
    public static void main (String args[]){...} }
class ChatClient {
    public static void main (String args[]){...} }
```

- Compilation

```
rmic monchat.Chat
```

RMI

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation

- Serveur de noms

- `rmiregistry` (méthodes `lookup()` et `bind()`)

- Lancement

- Serveur de noms (machine m1)

- `rmiregistry`

- Serveur (machine m2)

- `java -`

- `Djava.rmi.server.codebase=file:C:/Users/toto/RMI/src/serveur/ -Djava.rmi.server.hostname=192.168.1.38 -`

- `Djava.security.policy=server.policy serveur.Serveur`

- Client(machine m3)

- `java -`

- `Djava.rmi.server.codebase=file:/C:/Users/toto/RMI/src/client/ -Djava.security.policy=client.policy client.Client`

- `192.168.1.38`

RMI/IIOP

Plan

Généralités

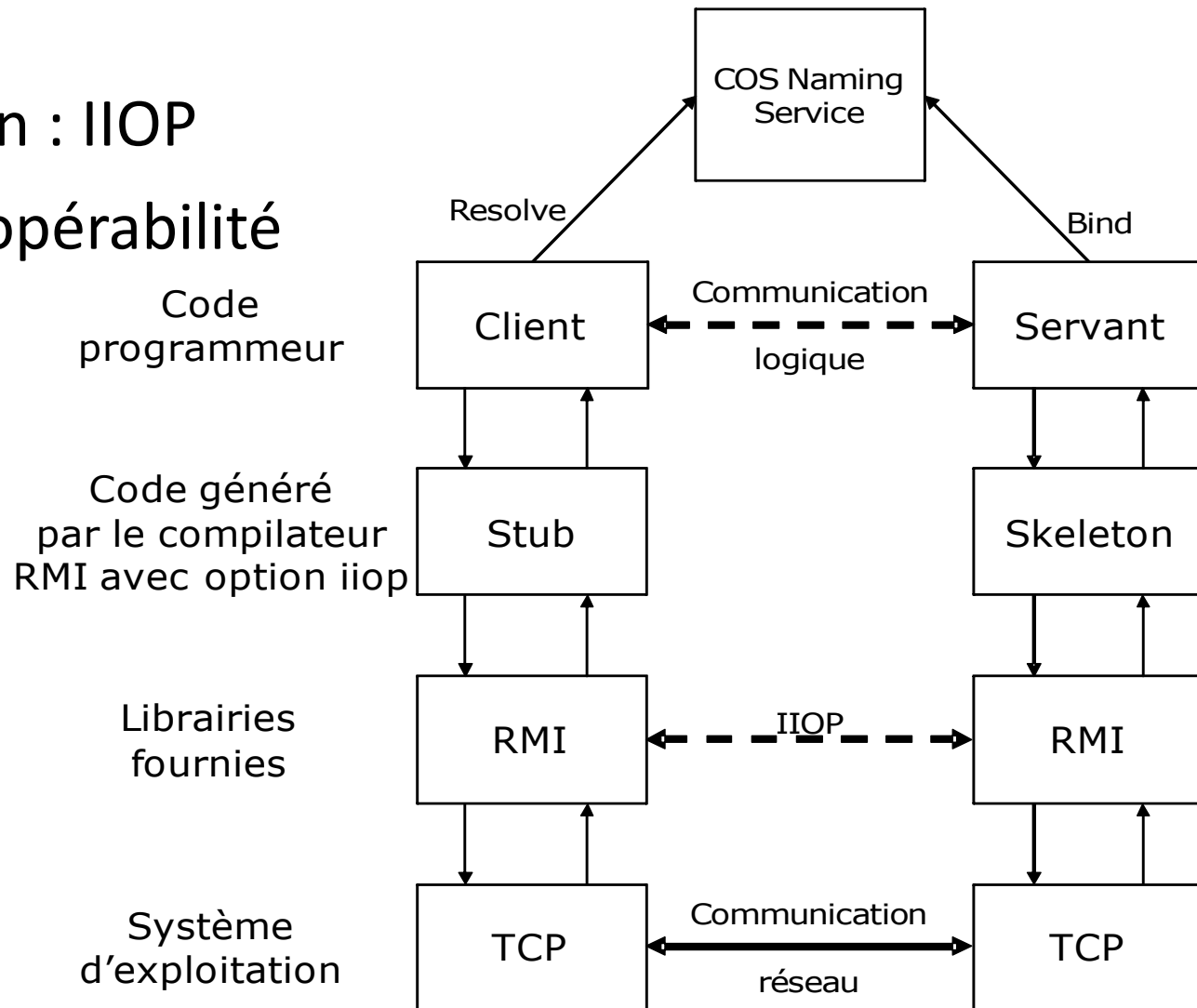
Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Protocole de communication : IIOP
- Permet l'interopérabilité RMI / CORBA



RMI/IIOP

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation (différences avec RMI)

- Implémentation du servant

```
class ChatServant extends PortableRemoteObject
    implements Chat {
    void setMessages (String auteur, String
    texte){...} }
```

- Transtypage complexe

- Compilation : `rmic -iiop monchat.Chat`

- Packages à importer

- Javax.rmi (servant, serveur, client)
 - Javax.naming (serveur, client)

RMI/IIOP

Plan

Généralités

Objets distribués hétérogènes

> Objets distribués homogènes

Synthèse objets distribués

Conclusion

- Exemple d'utilisation (différences avec RMI)

- Lancement

- Serveur de noms (machine m1)

- ```
tnameserv
```

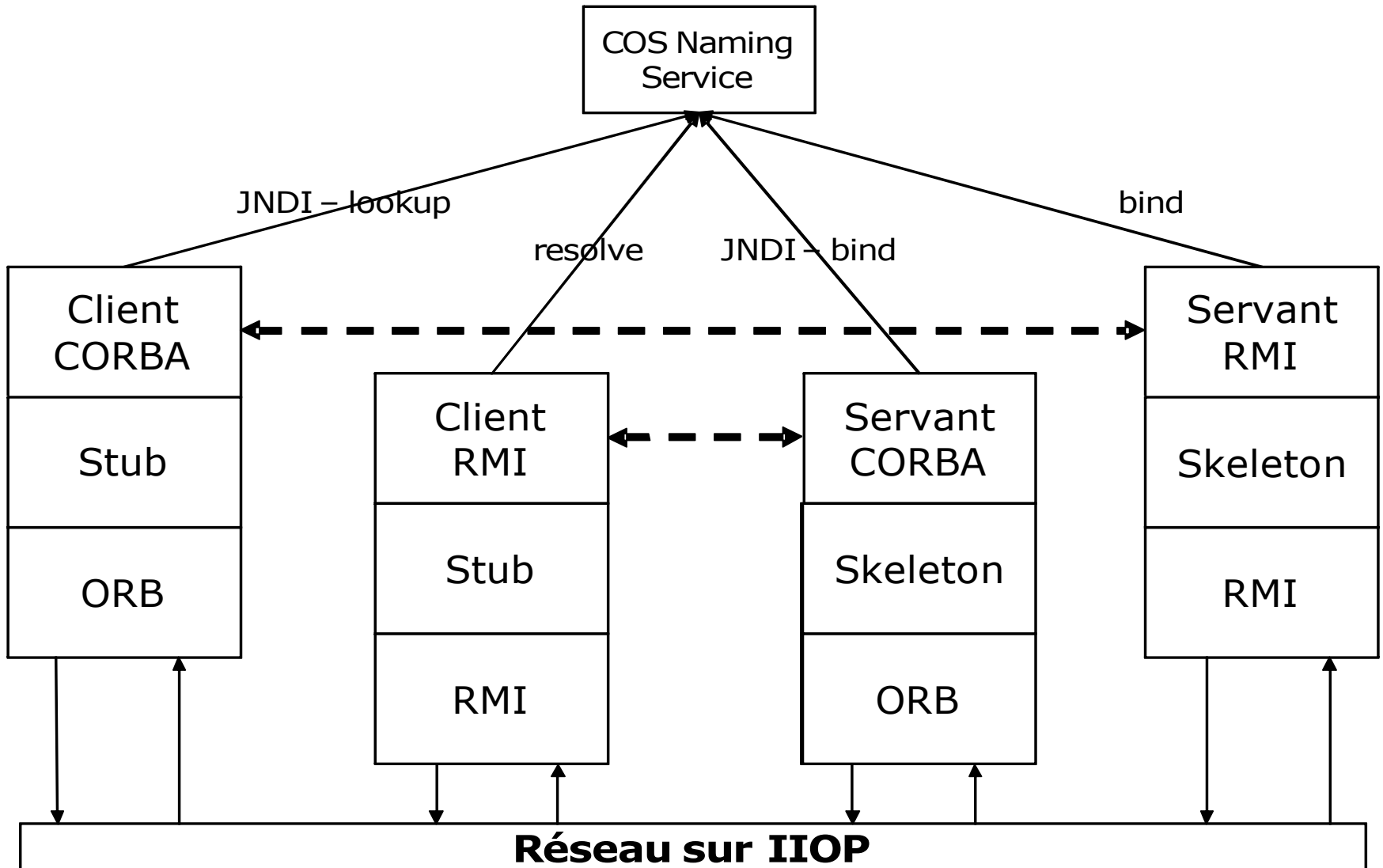
- Serveur (machine m2) :

- ```
java -Djava.rmi.server.codebase=http://m1/ ChatServer
```

- Client(machine m3) :

- ```
java -Djava.rmi.server.codebase=http://m1/ ChatClient
```

# CORBA + RMI/IIOP



# RMI : évolutions

## Plan

Généralités

Objets distribués hétérogènes

Objets distribués homogènes

> Synthèse objets distribués

Conclusion

- Dynamic classloading
  - Le serveur n'enregistre plus le stub, mais l'interface de l'objet et l'URL permettant de télécharger le stub et le skeleton
  - Le ClassLoader se charge de déployer les instances en fonction de l'URL au moment de la requête (lazy / just-in-time deployment)
- Suppression des skeletons (V 1.2)
  - Le stub du client possède suffisamment d'information pour contacter l'objet servant directement
  - Ajout du code manquant (Reflection) pour réaliser ce travail dans le stub
- Dynamic proxies
  - Génération dynamique (Reflection) de proxies implémentant les interfaces du serveur
  - Remplacement des stubs par des proxies dynamiques (V 1.5)...

# RMI : conclusion

- “RMI has been remarkably stable for a long time. Changes have been slow and incremental. But **each change has been towards simplifying development** and deployment, without any accompanying sacrifices. It's still a **strongly typed system**, it still **preserves** as much of the **local-process call syntax and semantics** as is reasonable, and it still is entirely Java. But, over time, it's also gotten **more agile**.
- Slowly and subtly, RMI has evolved into a low-process, low-deployment-overhead, and lightweight framework for strongly typed remote method invocation. It's become as good for very dynamic environments as any of the more loosely coupled frameworks without sacrificing any of its original strengths.”
- Source :  
<https://today.java.net/pub/a/today/2004/06/01/RMI.html>

# Conclusion

- Principes fondamentaux
  - Objets client, servant et serveur
  - Invocation d'objets distants transparente pour le client
  - Échange de messages conformes à des descriptions d'interfaces
  - Service de nommage
- Limites
  - Requiert la programmation du serveur
  - Requiert l'inscription dans le serveur de noms
  - Interfaces Générées à la compilation (CORBA non-DII)
- Mise en place « lourde » pour le développeur

# Conclusion

- Géré : accès aux services transversaux
  - Nommage (serveur de noms)
  - Transactions
  - Persistance
  - Sécurité...
- Non géré : optimisation des accès aux ressources
  - Pools de connexion ou de threads
  - Activation et désactivation des objets
  - Répartition de la charge
  - Tolérance aux pannes

# Références

## Plan

Généralités

Objets distribués hétérogènes

Objets distribués homogènes

Synthèse objets distribués

> Conclusion

- CORBA
  - <http://www.omg.org>
  - <http://corba.developpez.com/cours/>
  - <http://corba.developpez.com/presentation.htm>
- RMI
  - <http://docs.oracle.com/javase/8/docs/platform/rmi/spec/rmiTOC.html>
  - <https://today.java.net/pub/a/today/2004/06/01/RMI.html>
- RMI/IIOP
  - <http://docs.oracle.com/javase/7/docs/technotes/guides/rmi-iiop/>
- Exemples de code RMI et RMI/IIOP
  - <http://java.sun.com/developer/codesamples/index.html>
  - [http://thomasfly.com/RMI/rmi\\_tutorial.html](http://thomasfly.com/RMI/rmi_tutorial.html)
  - <http://www-128.ibm.com/developerworks/java/rmi-iiop/space.html>