Principes des Systèmes à Objets Répartis

Julien Sopena
Julien.Sopena@lip6.fr
(basé sur un cours de **Gaël Thomas**)

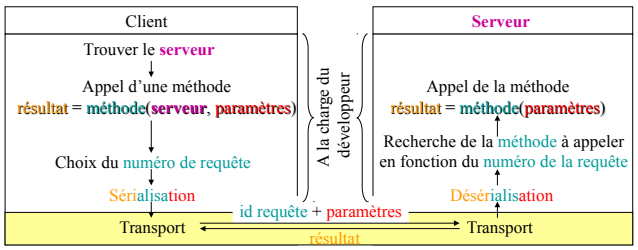
Université Pierre et Marie Curie
Master Informatique
M1 – Spécialité SAR

Principes des Systèmes à Objets Répartis

1. Interaction requête/réponse

Interaction requête/réponse
- Envoie d'une requête
- Traitement de la requête
- Envoie de la réponse

⇔ Appel de méthode à distance



2. Appel de procédures distantes

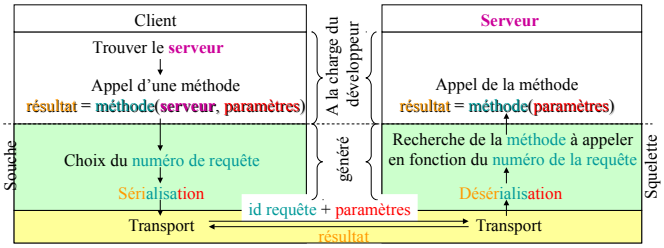
Objectifs d'un protocole d'appel de méthode à distance (par exp: RPC)
- Diminuer le travail de développement des serveurs et des clients
  - ✓ Prise en charge de la sérialisation/désérialisation des arguments
  - ✓ Prise en charge de la conversion appel de méthode en protocole requête/réponse
  - ✓ Plus facile pour un serveur d'offrir plusieurs services (méthodes)
- Masquer une partie de la répartition
  - ✓ Le client appel une méthode locale
  - ✓ Cet appel est délégué au serveur
  - ✓ La méthode est exécutée par le serveur
- Masquer l'hétérogénéité entre clients et serveurs
  - ✓ Format pivot pour les données (xdr, sérialisation Java, CDR…)

Comment : en générant le code de la délégation de l'appel
- Une souche d'appel est généré pour le client
- Un squelette du serveur est générée pour le serveur

2. Appel de procédures distantes

A la charge du développeur du client
- Trouver le serveur
- Appeler la méthode

A la charge du développeur du serveur
- Écrire la méthode



3. Modèle des objets répartis

Mais : appel de méthode inadéquat pour un langage objet
- Programmation fondamentalement impérative
  - ✓ Le serveur offre des méthodes, mais il est construit avec des objets
  - ✓ Le client appelle des méthodes, mais il utilise des objets
  - ⇒ Non transparent pour les développeurs
- La notion d'objet a été perdue

⇒ Introduction du paradigme des objets répartis
- Le serveur fournit un objet avec des méthodes et des champs
- Le client utilise un objet représentant l'objet serveur localement

3.a. Notion de mandataire

Notion (indépendante) de mandataire (proxy) :
- But : substituer un représentant à la place d'un objet qui offre les mêmes fonctionnalités (méthodes)
- Intérêt : adapter le code de l'objet de manière transparente pour l'utilisateur
  - ✓ Répartir le client et le serveur
  - ✓ Réutiliser des objets existants en leur ajoutant du code (avant/après)

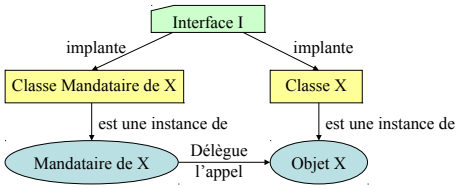
Principe : le client utilise un mandataire à la place de l'objet d'origine

Transparence pour le client

3.a. Notion de mandataire

Fonctionnement :
- Une classe X implante une interface I
- Le client utilise uniquement les méthodes de l'interface I
- Le mandataire doit implanter les méthodes de I

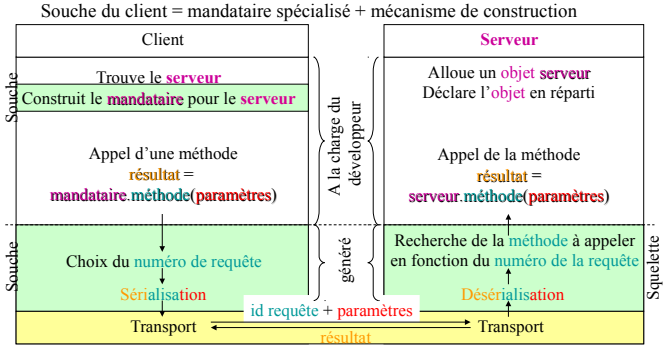
Schéma général de fonctionnement :



3.b. Paradigme des objets répartis

Paradigme des objets répartis

Souche du client = mandataire spécialisé + mécanisme de construction



4. Construction d'applications

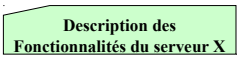
Construction d'une application à objets répartis

1. Définir l'interface de l'objet réparti dans un langage L0
   (Décrit les fonctionnalités du serveur)

1. Écrire le serveur dans un langage L1
   a. Développer un objet serveur implantant l'interface
   b. Écrire le serveur : allouer un objet serveur + exporter cet objet

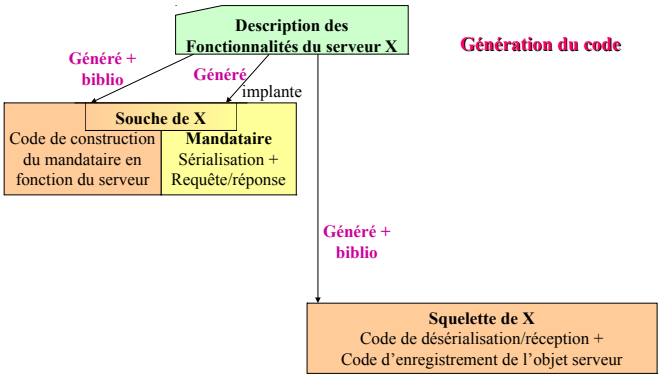
2. Écrire le client dans un langage L2
   a. Trouver le serveur
   b. Construire un mandataire à partir du serveur

4. Construction d'applications



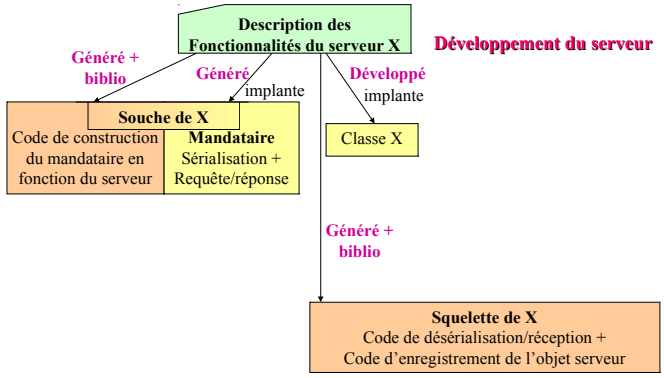
Développement de la description des fonctionnalités du serveur

4. Construction d'applications



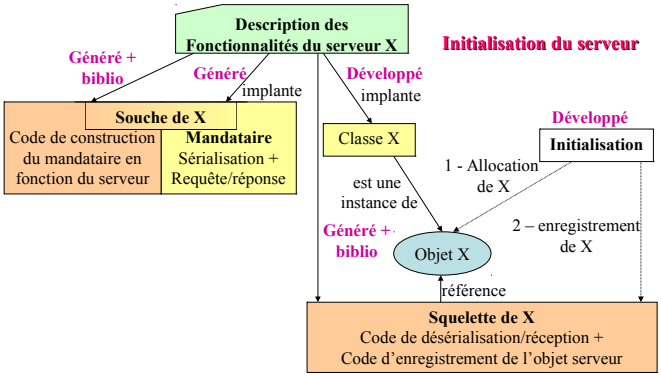
Génération du code

4. Construction d'applications



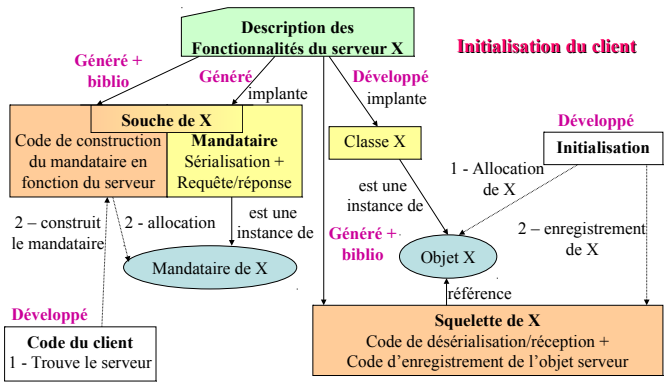
Développement du serveur

4. Construction d'applications



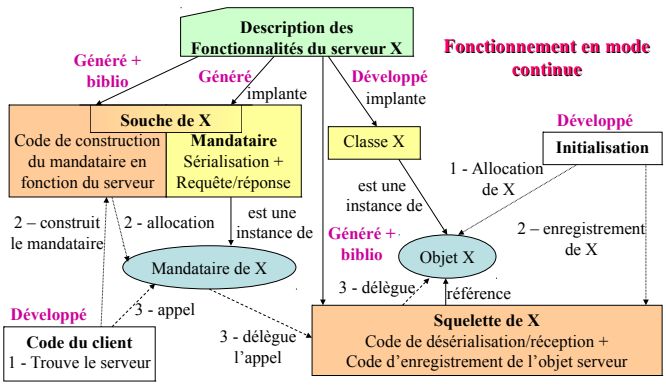
Initialisation du serveur

4. Construction d'applications



Initialisation du client

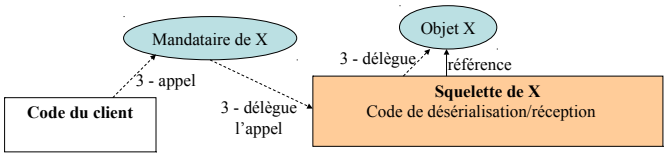
4. Construction d'applications



Fonctionnement en mode continue

4. Construction d'applications

Vue uniquement en mode continue



4. Construction d'applications

Remarque :

Le modèle à objets répartis n'impose pas que le client et le serveur soient homogènes

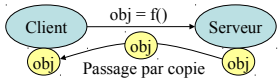
Langage, système et OS peuvent être différents

- ✓ La couche réseau masque la répartition
- ✓ La couche sérialisation masque l'hétérogénéité

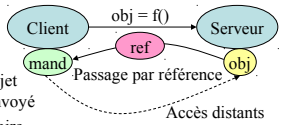
5. Passage de paramètres

Deux façons de passer les paramètres entre clients et serveurs

- Par copie
  - ✓ L'objet est copié lors de l'appel ou lors du retour de l'appel
  - ❖ La copie est l'original n'ont plus de rapport ⇒ si l'original est modifié, la copie de ne l'est pas



- Par référence
  - ✓ Une référence distante est envoyée et un mandataire est construit lors de l'appel ou lors du retour
  - ❖ Un objet passé par référence est un objet réparti! C'est un mandataire qui est envoyé
  - ❖ L'objet n'existe qu'en un seul exemplaire



5. Passage de paramètres

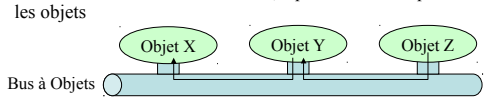
Passage par copie ou passage par référence?

- Par copie
  - \+ Copie locale ⇒ évite les accès distants
  - \- Copie ⇒ modification non répercutée sur l'original
  - \- Copie ⇒ pas forcément à jour si l'original est modifié
- Par référence
  - \+ Toujours accès à la dernière version
  - \+ Modification partagée par tous les utilisateurs du serveur
  - \- Accès distant à chaque accès à l'objet

6. Bus à objets

- **Bus à objets** : vue abstraite d'un système à objets répartis

  Constitué de l'ensemble des souches, squelettes sur lesquels sont enregistrés les objets



- En général, le bus à objets propose des objets systèmes (appelés **services systèmes**)
  - Service de résolution de nom : permet d'associer un nom à un objet réparti
  - Service de persistance : permet de sauvegarder l'état d'un objet
  - Service de transaction…
- **Intergiciel** (middleware) : couche intermédiaire entre l'application et le système (par exemple : bus à objets + services)

6. Bus à objets

Service de résolution de noms

Construction du mandataire nécessite

- L'adresse du serveur
- Un identifiant pour l'objet serveur

  Le serveur peut gérer plusieurs objets

⇒ **Notion de référence distante**

Définition : une **référence distante** identifie de manière unique un objet sur le bus à objets

La référence vers le mandataire est alors la référence locale

⇒ Le mandataire possède la référence distante de l'objet serveur

6. Bus à objets

Service de résolution de noms

Problème : comment trouver la référence distante d'un objet serveur?

Solution 1 : *à la main*

- Noter dans un fichier l'adresse du serveur

  Dépendant de la localisation physique du serveur
- Noter dans un fichier le numéro de l'objet serveur

  Peut changer à chaque exécution
- Charger dans le programme client le fichier et construire la référence distante

⇒ Pas satisfaisant car le serveur et le client doivent échanger directement des données avant de se connaître!

6. Bus à objets

Service de résolution de noms

Problème : comment trouver la référence distante d'un objet serveur?

Solution 2 : *utiliser un serveur de nom*

- Le serveur enregistre sous un nom connu la référence distante de son objet
- Le client interroge le serveur de noms

⇒ Le client a uniquement besoin de connaître le nom de l'objet serveur et le serveur de noms

La plupart des systèmes à objets répartis offrent un service de résolution de noms

- ✓ Implanté sous la forme d'un objet réparti (rmiregistry, cosnaming)
- ✓ Ou sous la forme d'une API connue (jndi)