# Callbacks distribués

Damien Olivier

Damien.Olivier@univ-lehavre.fr

Laboratoire d'informatique du Havre

# 1. Présentation

- Le callback distribué divise un appel synchrone qui retourne des informations en deux appels :

# 1. Présentation

- Le callback distribué divise un appel synchrone qui retourne des informations en deux appels :
  - Une requête (appel) ;

# 1. Présentation

- Le callback distribué divise un appel synchrone qui retourne des informations en deux appels :
  - Une requête (appel) ;
    - La requête est effectuée par un appel de méthode oneway de l'objet serveur en lui fournissant en entrée une référence d'objet distribué (le callback) pour que le serveur puisse lui renvoyer le résultat.

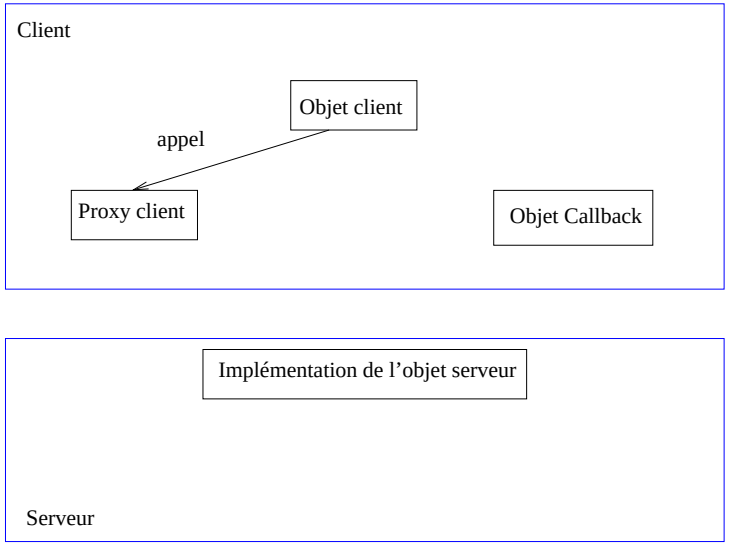
# 1. Présentation

- Le callback distribué divise un appel synchrone qui retourne des informations en deux appels :
  - Une requête (appel) ;
    - La requête est effectuée par un appel de méthode oneway de l'objet serveur en lui fournissant en entrée une référence d'objet distribué (le callback) pour que le serveur puisse lui renvoyer le résultat.
  - Une réponse (appel) ;

# 1. Présentation

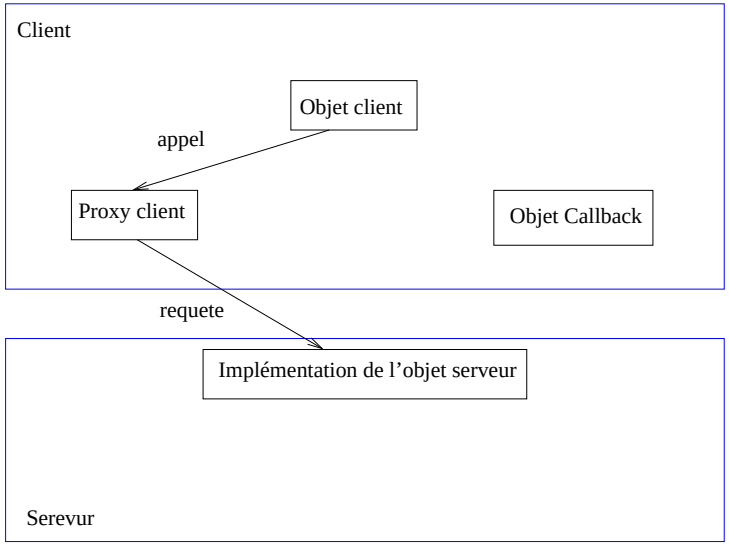
- Le callback distribué divise un appel synchrone qui retourne des informations en deux appels :
  - Une requête (appel) ;
    - La requête est effectuée par un appel de méthode oneway de l'objet serveur en lui fournissant en entrée une référence d'objet distribué (le callback) pour que le serveur puisse lui renvoyer le résultat.
  - Une réponse (appel) ;
    - Le résultat est transmis par l'intermédiaire de l'objet dont la référence a été transmise par le client.

# 1. Présentation



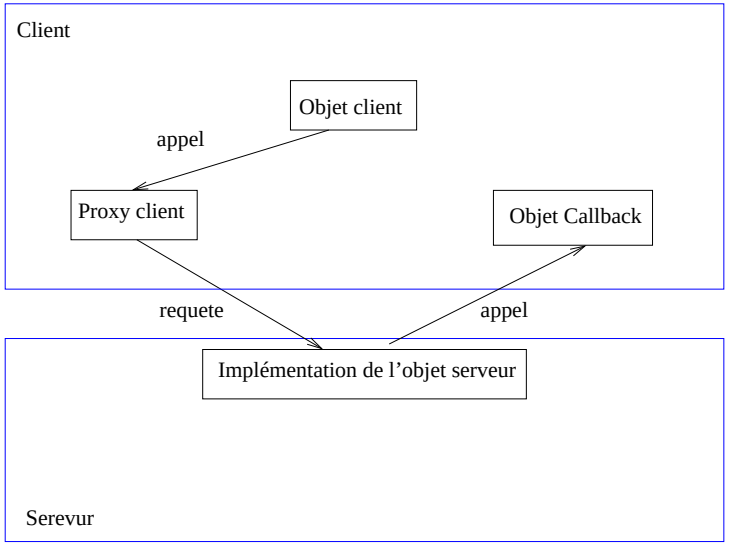

- Le client invoque une méthode et continue son exécution pendant que le serveur traite la réponse

# 1. Présentation



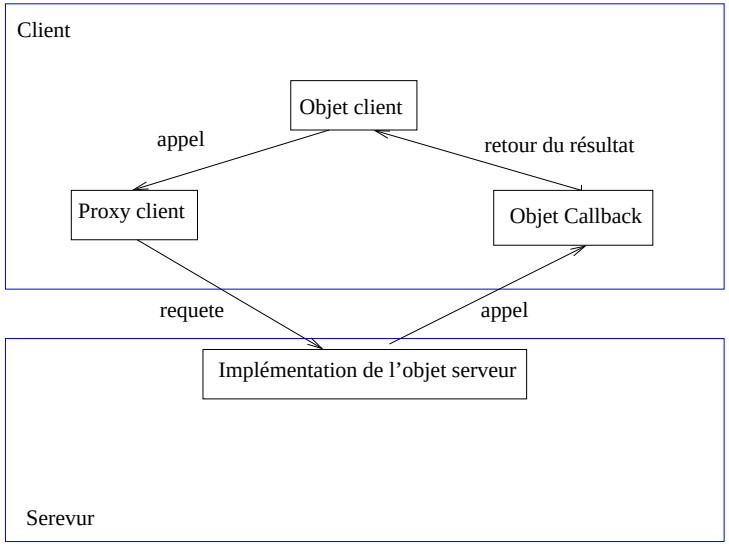

- Le client invoque une méthode et continue son exécution pendant que le serveur traite la réponse

# 1. Présentation




- Le client invoque une méthode et continue son exécution pendant que le serveur traite la réponse

# 1. Présentation




- Le client invoque une méthode et continue son exécution pendant que le serveur traite la réponse

# 2. Mise en œuvre

On définit deux interfaces :

- Une pour le serveur qui est implémentée par lui ;
- Une autre pour l'objet `callback` qui est implémenté par le client.

# 2. Mise en œuvre

## Définition des interfaces idl

- Coté serveur, l'interface
  - Les méthodes sont oneway et de type `void`

```
interface Calcul
{
 oneway void  add(in float operande1, in float operande2, in Calcul_callback client);
 oneway void mult(in float operande1, in float operande2, in Calcul_callback client);
};
```

- Coté client, l'interface
  - Le callback doit contenir toutes les informations nécessaires pour les valeurs de retour.

```
interface Calcul_callback
{
     oneway void  add_callback(in float resultat, in long status);
     oneway void mult_callback(in float resultat, in long status);
};
```

# 3. Quand doit-on utiliser un callback

- Lorsque le client gère une ou plusieurs tâches concurrentes, comme la gestion de l'interface et l'invocation du middleware ;
- Quand le traitement du serveur est long et que l'application cliente ne peut pas attendre ;
- Quand le client fait appel à plusieurs services simultanément et peut don recevoir des résultats à n'importe quel moment et dans un ordre quelconque ;
- Quand il est nécessaire au client de continuer un traitement local en même temps qu'il envoie des requètes Corba ;
- Quand un client accepte des résultats qu'à des intervalles de temps spécifiques.

# 4. Exemple développé

- Définition des deux interfaces `Calcul.idl` et `Calcul_callback.idl` ;
- Projection des interfaces ;
- Implémentation des services `Calcul_callback_impl.java`, et `Calcul_impl.java` ;
- Implémentation du serveur `Serveur.java` ;
- Implémentation du client `Client.java`.