INP ENSEEIHT

Correction de l'examen d'Intergiciels

Documents autorisés : tout document mis à disposition

Deuxième année Informatique et Réseaux

Durée : 1H45

2 octobre 2012

1 Processus communicants et intergiciels (12 points)

Les question suivantes abordent des points généraux sur les processus communicants et les intergiciels.

Questions (1,5 points par question)

1. Quel service de base est nécessaire dans tous les intergiciels pour permettre la désignation des entités à distance : procédures, méthodes, files, thèmes, etc ?

   Réponse *Les service de base est un service de nommage permettant de convertir un nom généralement symbolique en un nom interne. Le service doit être accessible en permanence et « localisable » par un moyen n'utilisant pas, bien évidemment le service lui-même (problème de récursivité). Par exemple, un serveur de nom Corba ou RMI sera repéré par le couple (nom de machine et numéro de port) et sollicité via l'interface de programmation par sockets.*

2. Pour quelle raison essentielle distingue-t-on plusieurs types de sémantiques pour l'appel de procédure à distance ?

   Réponse *Le mécanisme d'appel de procédure à distance est beaucoup moins fiable que le mécanisme d'appel local. En effet, les messages échangés nécessaires à l'implantation du mécanisme peuvent être perdus, dupliqués, ré-émis au bout d'un délai de garde, etc. Par ailleurs, le processus auquel est destiné l'appel peut être surchargé, en panne, et dans l'incapacité de satisfaire la requête. Les sémantiques proposées se distinguent par leur capacité plus ou moins forte à résister aux erreurs issues de la communication ou du serveur appelé. Quelle que soit la sémantique retenue, l'échec d'un appel est possible et une exception doit être levée dans ce cas chez l'appelant.*

3. Comment la transparence d'accès à un objet distant est implantée en Java ? Expliquer le schéma d'implantation en l'occurrence que référence le client ? que fait le serveur ?

   Réponse *La transparence d'accès est réalisée en introduisant un objet « proxy » côté client et un objet squelette côté serveur. L'objet proxy permettra de localiser l'objet réel concerné via un serveur de noms. Ce serveur de noms est lui-même un objet accessible à distance.*
   Le client référence donc un objet local proxy qui encapsule la transaction de messages implantant l'appel de méthode à distance. Le serveur retrouve le squelette et l'objet concerné par un appel. L'exécution de la méthode est invoquée via le squelette assurant la désérialisation du message d'appel et la sérialisation du message réponse.

4. Quel problème pose les objets accessibles à distance vis-à-vis d'un ramasse-miettes ?

   Réponse *Les objets accessibles à distance pose le problème des références « distantes ». Un objet accessible à distance peut très bien ne plus être référencé par les variables locales du serveur de l'objet alors que des « clients » possèdent encore des références distantes à cet objet. Par exemple, si l'objet a été enregistré dans un serveur de noms, cette référence existera tant que le serveur de noms sera actif. Il faut donc mettre en œuvre un algorithme spécifique pour ce genre d'objet nécessitant de gérer un compteur des références distantes à l'objet.*

5. Enoncer d'une part les points communs, d'autre part, les différences essentielles entre les modèles d'exécution proposés par le modèle `Linda` et la spécification `JMS`

   Réponse *Les deux modèles ont en commun l'usage d'un serveur intermédiaire, l'un gérant un espace de tuples dans* `Linda` *et l'autre, appelé courtier, gérant un espace de messages triés par thèmes ou insérés dans des files d'attente. Par contre, il se distinguent par :*
   – *les types d'objets qu'ils manipulent : des tuples pour* `Linda` *et des messages pour* `JMS`*.*
   – *les règles de désignation : approche associative par le contenu dans le cas* `Linda` *et approche par thèmes et files dans le cas* `JMS`*. En particulier, une requête* `Linda` *peut conduire à extraire un ensemble de tuples satisfaisant un ensemble de critères.*
   – *l'approche messages et donc ressources consommables de* `JMS`*.*
   De façon plus globale, le modèle `Linda` *est plus général dans la mesure où il permet d'implanter le modèle* `JMS` *assez simplement mais l'inverse n'est pas vrai.*

6. Comment rend-on persistant les thèmes (topics) ou files (queues) dans le modèle JMS ?

   Réponse *De façon très classique en enregistrant les thèmes ou files dans une base de données.*

7. Les intergiciels implantent différents modèles de calculs répartis. Parmi les familles d'intergiciels suivantes : CORBA, Linda, MOM-JMS et Java RMI, vues en cours, précisez laquelle ou lesquelles répondent aux propriétés suivantes :
   – Modèle orienté objet accessible à distance ;
   – Modèle offrant le protocole de communication de type publication/abonnement (publish/subscribe) ;
   – Modèle traitant l'hétérogénéité.

   Réponse

Modèle	*CORBA*	*Linda*	*MOM-JMS*	*Java RMI*
orienté objet à distance	×			×
publication/abonnement		×	×	
traitement de l'hétérogénéité	×			×

   Remarque *Pour le traitement de l'hétérogénéité, les deux environnements ont des approches radicalement différentes.*

8. Quel rôle joue le protocole IIOP dans la spécification de l'environnement Corba ?

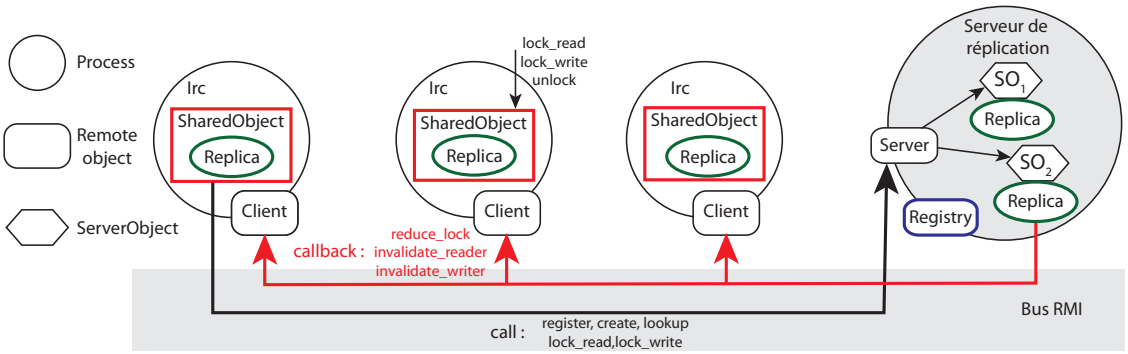
   Réponse *Le protocole IIOP (Internet Inter-ORB Protocol) est une spécialisation dédiée à l'Internet du protocole GIOP (General Inter-ORB Protocol). Le rôle est le même dans les deux cas : spécifier le protocole de communication entre intergiciels Corba développés par différents éditeurs de logiciels. En effet, une requête émise sur le bus logiciel ORB en utilisant une implantation donnée, par exemple IBM doit pouvoir être traitée par une implantation autre, par exemple SUN.*

2 Bus à objets répliqués : conception revisitée (8 points)

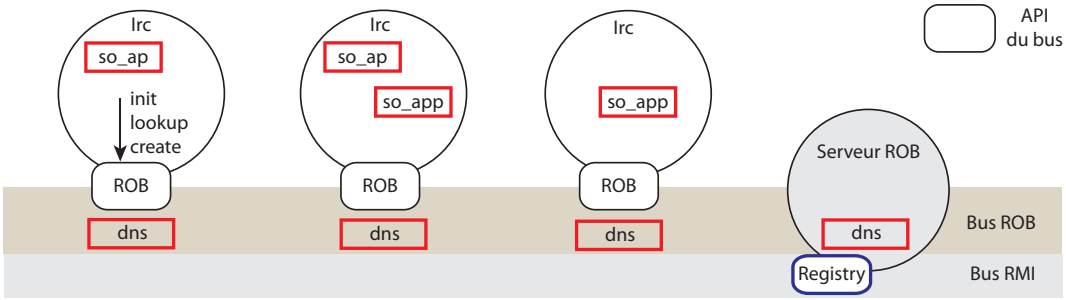
Le projet réalisé consistait à implanter un bus à objets répliqués en utilisant comme support un bus à objets accessibles à distance via le protocole Java RMI. L'objectif est ici de revisiter cette conception et de concevoir une solution un peu différente.

La solution proposée comporte en effet un serveur assimilable à un serveur de noms indispensable dans la plupart des intergiciels. Ce serveur assure en particulier la correspondance entre un nom externe d'objet répliqué et son moniteur de synchronisation (classe ServerObject).

La figure suivante rappelle l'architecture générale de cette conception :



Conception revisitée Nous avons pu constater que le service de nommage des intergiciels implantant l'appel de méthode à distance est lui-même considéré comme un objet accessible à distance (Registry pour Java RMI et NamingContextExt pour Corba par exemple). Par analogie, on peut donc envisager d'implanter le serveur de noms du bus à objets répliqués comme, lui-même, **un objet répliqué**. La figure ci-dessous illustre la démarche :



La classe ROB (Replicated Object Bus) implante l'API d'usage du bus. Cette API devrait rester très similaire à celle de la classe Client avec les méthodes statiques : init, lookup et create. L'objet répliqué dns de classe NameServer assure donc, quant à lui, un service de nommage entre tout nom externe d'objet répliqué et son moniteur de synchronisation. On propose (sans obligation) les interfaces suivantes :

```
public class ROB {
  public static void init() ;
  public static SharedObject lookup(String name) ;
  public static SharedObject create(String name, Object o) ;
}
public class NameServer {
  ServerObject lookup(String name) ;
  void register(String name, ServerObject monitor) ;
}
```

Question (8 points) : Vous devez décrire la conception générale de cette nouvelle implantation. Votre réponse est donc assez « ouverte » mais devra comporter les éléments de réponse suivants :

- la faisabilité de cette approche : soulève-t-elle des difficultés ou simplifie-t-elle au contraire l'implantation d'une solution ? Préciser, en particulier, qui crée l'objet répliqué « serveur de noms » et comment peut-il être repéré au niveau de la gestion du bus à objets répliqués ?
- la localisation du moniteur de synchronisation (ServerObject)de chaque objet répliqué applicatif créé.
- l'architecture générale de cette nouvelle solution. Décrire les interfaces et classes réutilisées, modifiées et/ou nouvelles de votre solution et préciser quelles classes sont accessibles à distance. Vous pouvez vous appuyer sur le schéma proposé en y apportant toute information supplémentaire jugée utile.
- les avantages et inconvénients d'une telle approche.