STM1CA2005

Examen Construction d'Applications Réparties

M1 - Master Sciences et Technologies Sem. 1
Mention Informatique

Université des Sciences et Technologies de Lille

2004-2005

3 heures – Tous documents autorisés

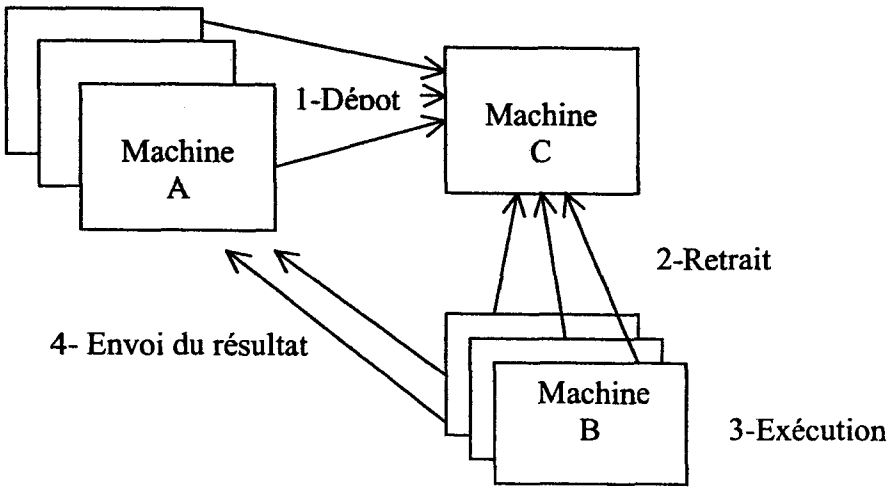
1. Partage de charge en CORBA (12 points)

L'application répartie suivante a pour objectif de permettre une répartition de la charge de travail entre plusieurs machines. Cet exercice examine étape par étape la conception de cette application. Nous vous demandons de répondre aux questions dans l'ordre où elles sont posées.

Trois types de machines A, B, C coopèrent en vue d'effectuer des calculs :

a. Les machines de type A déposent des demandes de travaux (un calcul par exemple) à l'aide d'une méthode `depot` auprès des machines de type C. Une exception permet de savoir si le dépôt ne se déroule pas correctement. Une demande de travaux peut posséder des paramètres,
b. Les machines de type B retirent des demandes de travaux à l'aide d'une méthode `retrait` auprès des machines de type C. Une exception permet de savoir si aucun travail n'est disponible. Les machines de type B effectuent le travail demandé et retourne le résultat à la fin de l'exécution à la machine A,
c. La machine de type C est le lieu de dépôt et de retrait des demandes de travaux.

Nous pouvons schématiser l'architecture de l'application et son fonctionnement de la façon suivante :



1. Pour chacune des trois machines de type A,B,C, dire si elle est cliente (de qui), serveur (pour qui) ou cliente/serveur (de qui et pour qui). (1 point)

Interfaces IDL

2. Définir l'interface en IDL CORBA de l'un des objets suivant installé sur les machines de type B. Cet objet nommé `Circle` propose deux méthodes permettant (1 point):

   - de déterminer la surface d'un cercle à partir d'un paramètre `rayon` donné sous la forme d'un réel,
   - de déterminer la circonférence d'un cercle à partir d'un paramètre `rayon` donné sous la forme d'un réel.

3. Définir l'interface en IDL CORBA de l'objet `Machine_C` installé sur la machine de type C permettant le dépôt et le retrait des demandes de travaux. Donner en français la signification précise de chaque élément défini (méthodes, paramètres, résultats, exceptions) (1 point).
4. Définir l'interface IDL `Resultat` de l'objet installé sur les machines de type A permettant de prendre en compte le résultat des travaux demandés (0,5 point).

Machine de type C

5. Définir la structure de données interne à l'objet de la machine C permettant de gérer le retrait et le dépôt des travaux. Commenter cette structure (0,5 point).
6. Les machines de type A et les machines de type B accèdent de façon concurrente à cette structure de données. Quels sont les risques ? Comment allez-vous gérer cet accès? (0,5 point)
7. Ecrire le code Java de la classe `Machine_CImpl` (1,5 point)

Machine de type A

La machine de type A souhaite être notifiée régulièrement de l'état des travaux qu'elle a demandé. Périodiquement la machine de type B exécutant le travail demandé informe la machine de type A initiatrice de la demande que le travail est en cours, en attente ou terminé. On suppose que ces notifications sont des informations non prioritaires dont la perte est sans conséquence.

8. Définir en IDL CORBA, l'interface `Notification` comprenant cette fonctionnalité. (0,5 point)
9. Donner l'interface complète de la machine de type A nommée `Machine_A` (0,5 point)
10. Les machines de type A doivent pouvoir à la fois envoyer des demandes de travaux vers les machines de type C et recevoir des résultats venant des machines de type B. Décrire en français la manière dont vous allez structurer le code pour permettre à ces machines de type A un fonctionnement dual. Vous préciserez l'ordre et/ou le parallélisme d'exécution des différentes parties du code. (1 point)

Généralisation des travaux demandés

Les machines de type B accueillent maintenant une bibliothèque mathématique importante. L'informaticien doit revoir l'application en vue de permettre l'invocation des travaux de façon générique, c'est-à-dire que les machines de type A doivent pouvoir soumettre.

11. Proposez un moyen de rendre générique les travaux déposés et retirés sur la machine C, c'est-à-dire un format de dépôt qui inclut la requête soumise et tous ses paramètres, quel que soit leur nombre et leur type. Vous utiliserez au mieux les moyens fournis par CORBA (1 point)

12. Comment les machines de type B vont-elles pouvoir reconnaître le type des paramètres soumis ? Donnez un exemple (1 point)

Questions de synthèse

13. Certains types de middleware vus dans le dernier cours fonctionnent selon le schéma décrit ci-dessus. Nommez les. (1 point)

14. Donnez les avantages et les inconvénients de ce type de fonctionnement. (1 point)

2. Détection et traitement de pannes au-dessus d'un protocole de transport en mode message (9 points)

Le but de cet exercice est de concevoir un protocole client/serveur qui traite différents cas de pannes pouvant survenir dans un environnement distribué. On rappelle qu'un protocole client/serveur met en relation deux entités (le client qui demande un service et le serveur qui exécute un traitement suite à ce service et fournit une réponse) et deux messages (le message d'appel envoyé par le client au serveur et la réponse du serveur au client). On se place dans l'hypothèse où la couche de transport est non fiable (aucune garantie quant à la livraison des messages – par exemple UDP) et ne fournit aucun mécanisme de détection, notification, ou correction de pannes. Il s'agit donc d'intégrer ces éléments au protocole client/serveur.

Questions générales

15. Dans le cadre d'un protocole client/serveur, quels types de panne peut-on rencontrer ? (1 point)
16. En termes de programmation, quels sont les mécanismes qui permettent de détecter une panne ? (1 point)

Traitement des pannes par un client

17. On suppose maintenant que le client fonctionne sans panne. Quels sont les différents problèmes que le client peut rencontrer lors de l'envoi du message d'appel vers le serveur ? Faire un schéma. (1 point)
18. Le client peut-il faire la distinction entre ces différents problèmes ? Proposer une solution pour les traiter. (1 point)

Traitement des pannes par un serveur

19. On suppose que le serveur fonctionne normalement. Quels sont les différents problèmes que l'on peut rencontrer lors de l'envoi de message de retour ? Proposez une solution pour les traiter. (1 point)

20. En supposant les problèmes des questions 17, 18 et 19 résolus, on veut maintenant pouvoir détecter les pannes survenant pendant le traitement de la requête. Quels sont les problèmes à régler ? Proposer deux solutions différentes. (2 points)

En Java, SO_TIMEOUT est le délai en millisecondes pendant lequel receive() attend un datagramme entrant avant de lever une InterruptedIOException Sa valeur ne doit pas être négative si SO_TIMEOUT vaut 0, receive() n'expire jamais; cette valeur peut être modifiée à l'aide de la méthode setSoTimeout() et lue avec la méthode getSoTimeout().

Exemple :

```
try{
      byte [] tampon = new byte[2056];
      DatagramPacket dp= new DatagramPacket(tampon, tampon.length);
      DatagramSocket ds= new DatagramSocket(2048);
      int expiration = ds.getSoTimeout();
      System.out.println
            (ds + "expirera apres" + expiration +"millisecondes");
      ds.setSoTimeout(30000); // attend 30secondes max
      expiration = ds.getSoTimeout();
      System.out.println
            (ds + "expirera apres" + expiration +"millisecondes");
      try
      {
          ds.receive(dp);
          // traite le paquet ...
      }
      catch (InterruptedIOException e)
      {
      ds.close();
      System.err.println("Aucune connexion dans les 30 secondes");
            }
      }
      catch (SocketException e) {
            System.err.println(e);
      }
      catch (IOException e){
            System.err.println("IOException inattendue "+e);
      }
```

21. Faire l'implantation d'une classe Java `ReliableSendRec` qui propose des méthodes d'envoi et de réception de messages fiables (2 points).