Université Lille 1
Master mention Informatique – M1
Module : Conception d'applications réparties (CAR)

Juin 2010

Examen

Conception d'applications réparties (CAR)

Tous documents papier autorisés. Téléphones, calculatrices et ordinateurs interdits.
Le barème est donné à titre indicatif. 3 heures.
Ce sujet comporte 3 pages.

1 Question de cours (2 points)

1.1 Soit une classe Java `Etudiant` héritant de la classe `Personne`. Expliquer en français ce que doit faire le programme serveur qui crée une instance de la classe `Etudiant` et la met à disposition pour qu'elle soit accessible via RMI. Même question pour CORBA.

1.2 À quoi correspond la notion de port en WSDL.

2 HTTP, servlet et REST (4 points)

2.1 On souhaite mettre en place en Java un serveur HTTP. Par rapport au serveur FTP vu en contrôle continu, expliquer en français les changements majeurs que cela implique. (1 point)

2.2 On souhaite que le serveur HTTP puisse exécuter des servlets. Expliquer en français ce qu'il faut mettre en place pour cela. (1,5 point)

2.3 On souhaite que le serveur HTTP, via une servlet, puisse servir de serveur REST, c'est-à-dire puisse recevoir et transmettre des requêtes à une classe Java comportant des annotations JAX-RS. Expliquer en français ce qu'il faut mettre en place pour cela. (1,5 point)

3 Calcul d'arbre couvrant en mode message et en RMI (4 points)

On considère un ensemble d'objets organisés selon une topologie en graphe. Chaque objet a un numéro unique fixé lors de la création du graphe. Chaque objet connaît uniquement ses voisins immédiats. On souhaite mettre en place un algorithme distribué permettant de calculer un arbre couvrant de ce graphe, c'est-à-dire que l'on souhaite pouvoir faire en sorte que chaque objet connaisse un père et des fils et que la structure distribuée résultante soit un arbre. L'algorithme doit pouvoir être lancé de n'importe quel site.

3.1 Peut-on supposer que, dès lors que l'on fixe l'objet à partir duquel l'algorithme est lancé, cela soit toujours le même arbre couvrant qui soit construit ? Justifier votre réponse. (0,5 point)

Stratégie asynchrone

3.2 Définir en français un ensemble de messages et en pseudo-code un algorithme permettant de calculer un arbre couvrant. (1,5 point)

Stratégie synchrone

3.3 Définir en Java, la ou les interfaces RMI ainsi que la ou les classes, permettant de calculer un arbre couvrant. Vous ferez en sorte que les objets puissent être *multi-thread* et que le calcul se déroule de façon concurrente sur plusieurs chemins de l'arbre. (2 points)

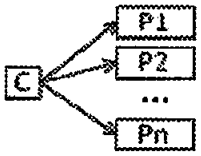
4 Protocole de validation à deux phases en CORBA (10 points)

Cet exercice a pour but d'étudier la conception d'un protocole transactionnel de validation à deux phases avec des objets CORBA.

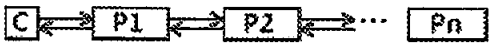
4.1 Rappeler la définition d'un protocole de validation à deux phases vu en cours. (1 point)

Centralized 2PC

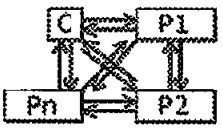
Dans un premier temps, nous étudierons la version dite *Centralized 2PC* de ce protocole (voir figure ci-dessous). Dans cette version, on considère qu'un objet coordinateur, noté C, dispose à un moment donné de la référence de tous les objets participants à la transaction. Les participants sont en nombre quelconque et sont notés P1, P2, ... , Pn.



Centralized 2PC



Linear 2PC



Distributed 2PC

Dans cette version, trois interfaces sont définies : `TransactionItf`, `GestionItf` et `ParticipantItf`.

L'interface `TransactionItf` permet à un objet utilisateur noté U d'effectuer une transaction. Elle comporte les opérations suivantes :

- `begin` : demande de démarrage d'une transaction. Retourne un entier représentant l'identifiant de transaction.
- `commit` : demande de validation d'une transaction. Prend en paramètre un identifiant de transaction. Retourne un booléen pour indiquer si la transaction a pu être engagée.
- `rollback` : demande d'annulation d'une transaction. Prend en paramètre un identifiant de transaction. Retourne un booléen pour indiquer si la transaction a pu être annulée.

L'interface `GestionItf` permet d'enregistrer un participant auprès du coordinateur. Elle comporte les opérations `register` et `remove` prenant chacune en paramètre un participant et un identifiant de transaction et permettant respectivement d'enregistrer et de retirer un participant.

L'interface `ParticipantItf` est une interface permettant d'identifier un participant et ne comporte pour l'instant aucune opération.

4.2 Définir en CORBA IDL les interfaces `TransactionItf`, `GestionItf` et `ParticipantItf`. (1 point)

4.3 Parmi les objets U, C, P1, P2, ... Pn, indiquer quels objets implémentent quelle(s) interface(s). (0,5 point)

On considère une transaction avec des opérations `credit` et `debit` permettant de créditer et débiter un compte bancaire d'un certain montant. Ces opérations sont définies dans une interface `BanqueItf` qui étend `ParticipantItf`. Elles permettent de réaliser une transaction qui correspond au transfert d'un certain montant entre deux comptes bancaires.

On s'intéresse à la conception du protocole client/serveur permettant de mettre en œuvre une telle transaction. Pour cela, vous pourrez être amené à étendre les interfaces définies ci-dessus. Votre solution doit faire en sorte de respecter les contraintes suivantes :

1. Le coordinateur doit conserver un comportement général et ne pas comporter de spécificité due au fait que la transaction correspond à un transfert entre comptes bancaires.
2. Le coordinateur ne connait pas a priori les participants. Ceux-ci doivent donc lui être indiqués par l'utilisateur pour chaque transaction.

4.4 Proposer une solution permettant de mettre en œuvre une telle transaction. Votre présentation peut être accompagnée de quelques phrases expliquant les choix effectués, d'un diagramme de séquence illustrant les invocations d'opérations entre les objets U, C, P1, P2, ... Pn et le cas échéant de la définition en CORBA IDL des nouvelles interfaces que vous proposez. (2 points)

4.5 Écrire en Java le code des objets coordinateur et participant. (1,5 point)

Linear 2PC

On s'intéresse maintenant à la version dite *Linear 2PC*. Comme illustré sur la figure, dans cette version, le coordinateur dispose à un moment donné de la référence du premier participant, qui lui-même dispose de la référence du participant suivant, etc. jusqu'au dernier participant.

4.6 Expliquer comment cette version permet de réduire le nombre d'invocations d'opérations en effectuant la transaction en une seule phase. (1 point)

4.7 Proposer une solution permettant de mettre en œuvre une telle transaction. Votre présentation peut être accompagnée de quelques phrases expliquant les choix effectués, d'un diagramme de séquence illustrant les invocations d'opérations entre les objets U, C, P1, P2, ... Pn et le cas échéant de la définition en CORBA IDL des nouvelles interfaces que vous proposez. (1,5 point)

Distributed 2PC

On s'intéresse maintenant à la version dite *Distributed 2PC*. Comme illustré sur la figure, dans cette version, le coordinateur et tous les participants disposent à un moment donné de la liste des références de tous les participants et du coordinateur.

4.8 Expliquer comment cette version permet d'effectuer la transaction en une seule phase. (0,5 point)

4.9 Proposer une solution permettant de mettre en œuvre une telle transaction. Votre présentation peut être accompagnée de quelques phrases expliquant les choix effectués, d'un diagramme de séquence illustrant les invocations d'opérations entre les objets U, C, P1, P2, ... Pn et le cas échéant de la définition en CORBA IDL des nouvelles interfaces que vous proposez. (1 point)