Intergiciels

1h45, documents autorisés

15 avril 2016

Les exercices sont indépendants.

1 Questions de cours (4 pt)

1. Pour quelle raison essentielle distingue-t-on plusieurs types de sémantiques pour l'appel de procédure à distance ?
2. Comment la transparence d'accès à un objet distant est-elle implantée en Java ?
3. L'intergiciel Linda est-il de type client-serveur, de type intergiciel à message, ou de type mémoire partagée ?
4. Citer au moins une fonctionnalité d'interaction aisée à mettre en œuvre avec un intergiciel à message (comme JMS) et difficile (voire impossible) avec Linda.

2 Sockets (5 pt)

On souhaite réaliser un répartiteur de charge (load balancer) qui reçoit des requêtes HTTP et les distribue de façon aléatoire vers un ensemble de serveur web.

Le répartiteur est configuré statiquement :

```
char *hosts[] = { "web1", "web2", "web3" };
int   ports[] = { 8081, 8081, 8082 };
int   nbhosts = 3;
```

A chaque réception d'une requête HTTP, le répartiteur de charge transfère la requête à un des serveurs web (les adresses de ces serveurs sont données par les tables hosts et ports) et en retour il transfère le résultat de la requête à l'émetteur. Le choix du serveur web est aléatoire (`random() % nbhosts` retourne un entier entre 0 et `nbhosts-1`).

Pour simplifier les entrées/sorties, que les requêtes et réponses sont lues ou écrites en un seul appel de `read/write` avec un buffer de 1024 bytes.

1. Donner la boucle principale du répartiteur.
2. Donner le code correspondant au traitement d'une requête.

3 RMI (7 pt)

On souhaite implanter un service de courtage qui interroge des banques pour trouver le meilleur emprunt (cf figure 1). Le client s'adresse au courtier (unique), le courtier interroge les banques, détermine la meilleure proposition et retourne au client cette proposition. Le client peut alors accepter cette proposition en s'adressant directement à la banque concernée. L'objectif est de réaliser ce système avec des objets accessibles à distance (RMI).

Les classes et interfaces (incomplètes vis-à-vis de Serializable et Remote) sont présentées figures 2 et 3. L'objectif est d'écrire un client comme celui de la figure 4.

1. Expliquez pourquoi la classe `Proposition` doit être Serializable.
2. Dans le code figure 4, combien d'appels à distance le client effectue-t-il et quels sont-ils ?
3. Complétez l'interface `Banque` pour en faire une interface d'objet accessible à distance.
4. On considère l'implantation (incomplète) de `Banque` de la figure 5. Complétez ce qui manque pour en faire une classe réalisant un objet accessible à distance.
5. Comment le service de courtage peut-il connaître les banques existantes ? Détaillez votre mise en œuvre.
6. Donnez le code de `CourtageImpl`, implantation de `Courtage`.
7. Comment le service de courtage peut-il découvrir qu'une banque a fermé ?
8. On ajoute une nouvelle fonctionnalité : suite à une première demande de proposition, une banque peut informer directement le client qu'elle a une nouvelle proposition à lui faire. Expliquez quelle(s) nouvelle(s) interface(s) est(sont) nécessaire(s), quelle(s) méthode(s) il faut modifier, et donnez la nouvelle architecture du système (en particulier schéma des objets accessibles à distance).

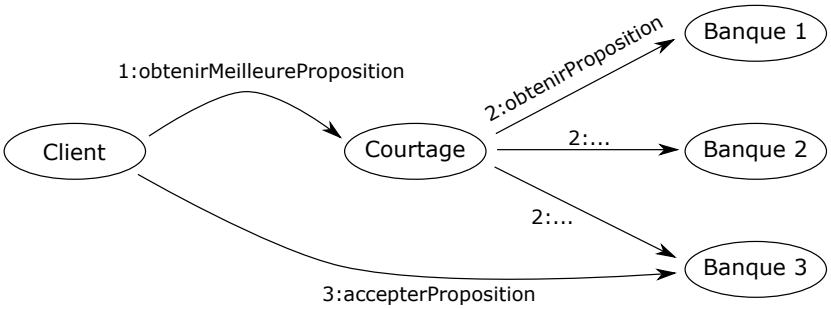

FIGURE 1 – RMI : Schéma général

```
import java.util.Date; // serializable
public class Proposition implements java.io.Serializable {
    private Banque banque;  /** Banque faisant cette proposition. */
    private double taux;    /** Taux proposé par la banque */
    private Date limite;    /** Limite de validité de l'offre */
    // + accesseurs getBanque(), getTaux(), setBanque(), setTaux()...
}
```

FIGURE 2 – Classe Proposition

```
public interface Banque {
    /** Nom de la banque. */
    String getNom();
    /** Demander une proposition de prêt pour sommeEmpruntee. */
    Proposition obtenirProposition (int sommeEmpruntee);
    /** Accepter une proposition. */
    void accepterProposition (Proposition proposition);
}

public interface Courtage {
    /** Interroge toutes les banques connues du service de courtage,
        et renvoie la meilleure proposition. */
    Proposition obtenirMeilleureProposition(int somme);
}
```

FIGURE 3 – Interfaces Banque et Courtage

```
import java.rmi.*;
public class Client {
  public static void main(String a[]) throws Exception {
    Courtage courtage = (Courtage) Naming.lookup("rmi://localhost:1099/MonCourtierFavori");
    Proposition p = courtage.obtenirMeilleureProposition(10000);
    System.out.println("La banque "+ p.getBanque().getNom() + " propose " + p.getTaux());
    p.getBanque().accepterProposition(p);
  }
}
```

FIGURE 4 – Classe Client

```
// import qui vont bien (sans importance)
public class BanqueImpl implements Banque {
    private String nom;
    private double taux;

    public BanqueImpl(String nom, double taux) { this.nom = nom; this.taux = taux; }

    public String getNom() { return nom; }

    public Proposition obtenirProposition (int sommeEmpruntee) {
        Proposition p = new Proposition(this, taux, new Date(2012, 07, 15));
        return p;
    }

    public void accepterProposition (Proposition proposition) {
        System.out.println ("Ma proposition à "+proposition.getTaux()+" a été acceptée.");
    }
}
```

FIGURE 5 – Classe BanqueImpl (incomplète)

4 Intergiciel à messages – JMS (4 pt)

On souhaite implanter un service de courtage qui interroge des banques pour trouver le meilleur emprunt (cf figure 6). Le service de courtage est réalisé par plusieurs courtiers. Le client s'adresse au service de courtage, *l'un* des courtiers prend en charge la requête du client, interroge les banques, détermine la meilleure proposition et retourne au client cette proposition. Le client peut alors accepter cette proposition en s'adressant directement à la banque concernée. L'objectif est de réaliser ce système avec un intergiciel à messages (JMS).

1. Indiquez quels sont les Destinations nécessaires, en précisant s'il s'agit de Topic ou de Queue.
2. Sachant qu'il peut y avoir plusieurs clients et plusieurs courtiers, expliquez comment on peut garder trace du lien entre la requête du client, les propositions formulées par les banques et la meilleure d'entre elles.
3. Décrivez alors la structure des messages échangés : pour chaque Destination, décrivez le contenu des messages envoyés.

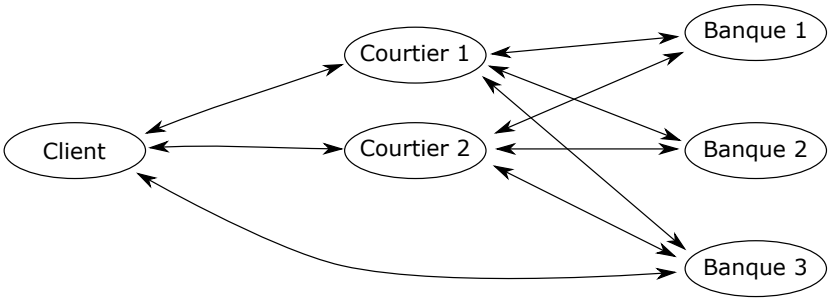

FIGURE 6 – MoM : Schéma général