Master Informatique 1
ALR: Architectures Logicielles Réparties

DS – Session d'Avril 2005

Denis Caromel, Christian Delbé, Romain Quilici

Université de Nice - Sophia Antipolis, Département Informatique

Les documents et calculatrices sont autorisés.
Les réponses ne sont pas longues, soyez *concis* et ... *lisible* !
Notez *clairement* sur votre feuille les numéros des questions (exemple: 2. f.).
** Les points sont donnés à titre indicatif **
Durée: 2H00

1 (2 points)

a.) Expliquez pourquoi TCP n'est pas un protocole adapté à la transmission de flux d' information associé aux applications temps réel interactives.

b.) Expliquez l'approche trés simple utilisée par le couple RTP/RTCP pour gérer au mieux ce problème.

2 (8 points)

a.) Quel est le rôle du couple **stub-skeleton** géneré par le compilateur rmic ? Peut-on se passer du stub ? Du skeleton ? Jusitifiez vos réponses.

b.) Qu'est-ce qu'un objet **activable** ? Quel est l'interêt de rendre un objet activable ?

c.) Quel est le rôle du **rmiregistry** en Java RMI ?

d.) Quelles sont les relations entre le **distributed garbage collector** (RMI) et le **garbage collector** standard ?

e.) Donnez en quelques mots le fonctionnement du pattern **smart proxy**. Quelle est son utilité ? Donnez un exemple d'utilisation.

f.) Quel est le rôle du **codebase** en Java RMI ?

g.) Quels sont les types d'interopérabilités offerts par **CORBA** ?

h.) Quelles sont les différences entre un **Session bean** et un **Entity Bean** dans le contexte des EJBs ?

3 (7 points)

Le but de cette partie est de comprendre le fonctionnement et le deploiement d'un programme RMI. Considérons les types suivants :

```
-------------------------- Type A ----------------------------

public interface RemoteA extends java.rmi.Remote {
    public void talk() throws java.rmi.RemoteException;
}


public class A extends java.rmi.server.UnicastRemoteObject implements RemoteA {

    public A() throws java.rmi.RemoteException {
        System.out.println("Create a new A on " + this.getCurrentHost());
    }
```

```
    public void talk()  throws java.rmi.RemoteException {
        System.out.println("A.talk() on " + this.getCurrentHost());
      }

    private String getCurrentHost() {
        try {
          return java.net.InetAddress.getLocalHost().getHostName();
        } catch (java.net.UnknownHostException e) {
          return "UNKNOWN";
        }
    }

    private void readObject(ObjectInputStream in) {
        try {
            System.out.println("A.readObject() on " + this.getCurrentHost());
              in.defaultReadObject();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    private void writeObject(ObjectOutputStream out) {
        try {
            System.out.println("A.writeObject() on " + this.getCurrentHost());
            out.defaultWriteObject();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

-------------------------- Type B ----------------------------

public class B implements java.io.Serializable {

    public B() {
        System.out.println("Create a new B on " + this.getCurrentHost());
    }

    public void talk() {
        System.out.println("B.talk() on " + this.getCurrentHost());
    }

    private String getCurrentHost() {
        try {
            return java.net.InetAddress.getLocalHost().getHostName();
        } catch (java.net.UnknownHostException e) {
            return "UNKNOWN";
        }
    }

    private void readObject(ObjectInputStream in) {
        try {
            System.out.println("B.readObject() on " + this.getCurrentHost());
            in.defaultReadObject();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
```

```
    private void writeObject(ObjectOutputStream out) {
        try {
            System.out.println("B.writeObject() on " + this.getCurrentHost());
            out.defaultWriteObject();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

On définit un serveur (classe `Server`) qui peut créer et utiliser des objets de type `A` et `B`.

```
public interface RemoteServer extends java.rmi.Remote {
    public RemoteA getNewA() throws java.rmi.RemoteException;
    public B getNewB() throws java.rmi.RemoteException;
    public void askAForTalking(RemoteA a) throws RemoteException;
    public void askBForTalking(B b) throws RemoteException;
}

public class Server extends java.rmi.server.UnicastRemoteObject implements RemoteServer {

    public static String SERVERNAME=''MyServer'';

    public Server() throws RemoteException{}

    public RemoteA getNewA() throws RemoteException {
        return new A();
    }

    public B getNewB() throws RemoteException {
        return new B();
    }

    public void askAForTalking(RemoteA a) throws RemoteException  {
        a.talk();
    }

    public void askBForTalking(B b) throws RemoteException {
        b.talk();
    }

    public static void main(String[] args) {
       // ****** A FAIRE ******
    }
}
```

Enfin, on définit un type client `Client`, qui va utiliser le serveur:

```
public class Client {

    private RemoteServer server;

    public Client() {}

    public static void main(String[] args) {
       // ****** A FAIRE ******
    }
}
```

a.) Quels sont les classes à compiler avec `rmic`? Quelles sont les classes générées?

b.) Ecrire la méthode `main()` de la classe `Server` qui crée un objet `Server` et l'enregistre dans le `rmiregistry` de la machine `MachineServeur.unice.fr` .

c.) Ecrire la méthode `main()` de la classe `Client` qui récupère une référence sur le serveur. Ce client sera démarré sur la machine `MachineClient.unice.fr`.

d.) Pourrait-on écrire une méthode `main()` uniquement dans le client demarré sur `MachineClient.unice.fr`, qui enregistrerait le serveur sur le `rmiregistry` de `MachineServeur.unice.fr` et récupèrerait une référence sur lui? Justifiez votre réponse.

e.) Donnez les traces produites *sur l'écran du serveur et sur celui du client* par chacun des appels suivants. Ces appels sont effectués séquentiellement depuis la methode `main()` du client, une fois que la référence `server` a été correctement initialisée.

```
public static void main(String[] args) {
  // INITIALISATION DE LA VARIABLE SERVER ...

  RemoteA a = server.getNewA(); // APPEL 1
  B b = server.getNewB(); // APPEL 2
  a.talk(); // APPEL 3
  b.talk(); // APPEL 4
  server.askAForTalking(a); // APPEL 5
  server.askBForTalking(b); // APPEL 6
}
```

4 (3 points)

a.) Dans le cadre d'une recherche et d'une utilisation d'un Service Web, citez les technologies utilisées (spécifiques aux Services Web) et leur rôle respectif.

Nous définissons le système suivant:

- Un client avec un PC sans configuration ni outil spécifique, et possédant une connexion Internet.
- "Dèle", une entreprise qui vend des ordinateurs par Internet.
- "Upéhesse", une entreprise qui offre un service de livraisons avec tracabilité.
- "Saufincau", une société qui propose des crédits.

Dèle offre au client la tracabilité de son colis (delegué à l'entreprise Upéhesse) ainsi qu'une possibilité de crédit sur l'achat (grâce à la banque Saufincau). Les liaisons suivantes existent entre les différentes entités :

- Le client achète un ordinateur chez Dèle depuis son domicile.
- Le client peut suivre la livraison de son achat grâce à l'entreprise Upéhesse.

Lors de l'achat, Dèle informe l'entreprise Upéhesse qu'une livraison doit être effectuée avec tracabilité, et informe Saufinco de la demande de crédit.

a.) Parmi les deux technologies **accès internet (HTTP seul)** et **accès Web Services**, donnez le mode de connection entre:

1. Client –> Dèle
2. Client –> Upéhesse
3. Dèle –> Upéhesse
4. Dèle –> Saufinco

b.) On se place maintenant à l'intérieur du système du vendeur Dèle qui possède 5 entités. Ce système nécessite :

- Une entité pour traiter les requêtes des clients, et transmettre certaines informations aux entres entités,
- Une entité pour envoyer les détails de la livraison à Upéhesse,
- Une entité pour envoyer les détails du crédit à Saufinco,
- Une entité qui stoque les informations du client dans l'entité base de données,
- Une base de donnée interne au vendeur de PC Dèle.

Proposez une architecture N-Tiers qui réponde aux contraintes ci-dessus, en spécifiant bien dans quel tiers se trouve chaque entité.