Université Joseph Fourier - année 2004-2005
RICM-3 ème année
Master 2ème année Professionnel Génie Informatique

Examen de "Systèmes et Applications Répartis"
13 décembre 2004

Durée : 2 heures ; tous documents autorisés

N.B. L'examen comporte 3 problèmes indépendants. Prière de lire attentivement l'ensemble de l'énoncé avant de commencer à répondre, et de respecter les notations du texte. La longueur de l'énoncé n'est pas un signe de difficulté, mais est nécessaire pour bien spécifier les problèmes. La **clarté**, la **précision** et la **concision** des réponses, ainsi que leur **présentation matérielle**, seront des éléments importants d'appréciation.

Problème 1. (5 points).

Les questions sont indépendantes. On demande des réponses **brèves** mais **précises**.

1. Expliquer le rôle d'un conteneur dans un système à composants.
2. Expliquer le fonctionnement d'une paire de processus dans un système à la Tandem Non-Stop. Expliquer précisément ce qui se passe en cas de panne d'un des processeurs.
3. Dans un système réparti, on souhaite conserver une donnée (par exemple sous forme de fichier) en plusieurs exemplaires sur des sites différents.
    a) Quel intérêt voyez-vous à cette duplication ?
    b) Par analogie avec ce qui a été vu pour les serveurs, expliquer quelles méthodes peuvent être utilisées pour assurer la cohérence de ces exemplaires multiples. Expliquer comment sont réalisées les opérations d'écriture et de consultation.
    c) Quelle est la méthode pour laquelle le temps de réponse d'une lecture est le plus petit ?

Problème 2 (6 points)

Un serveur sécurisé a pour objectif de permettre les échanges de type requête-réponse à l'intérieur d'une entreprise, en assurant leur confidentialité. La méthode adoptée est décrite ci-après.

On utilise un serveur central S sécurisé. Chaque client a une clé secrète, qui n'est connue que de lui et du serveur central S (tous les clients font confiance à S). Lorsqu'un client (soit A) désire envoyer une requête à un autre client (soit B), il procède comme suit :

A fabrique un nombre aléatoire R.

$A \rightarrow S : (A, B, \{R\}_{KA})$ (KA est la clé secrète de A, connue de A et de S)

$S \rightarrow A : \{R\}_{KB}$ (KB est la clé secrète de B, connue de B et de S)

$A \rightarrow B : (\{Requête\}_R, \{R\}_{KB})$

$B \rightarrow A : \{Réponse\}_R$

Comme B connaît KB, il peut déchiffrer $\{R\}_{KB}$, donc obtenir R et ensuite déchiffrer $\{Requête\}_R$ pour obtenir Requête, et enfin renvoyer la réponse Réponse, chiffrée par R.

Question 1. Montrer que ce protocole n'est pas sûr, en donnant avec précision (indiquer tous les messages échangés et les actions effectuées) un scénario dans lequel un espion X "de l'intérieur" (c'est-à-dire ayant un compte dans l'entreprise) peut violer la confidentialité de l'échange ci-dessus entre A et B. On suppose que X peut intercepter et renvoyer des messages sur le réseau.

Question 2. Quelles modifications pouvez-vous proposer pour rendre sûr ce protocole, en supposant toujours possibles l'interception et le rejeu de messages (plusieurs solutions sont possibles) ?

Indication : noter que le serveur S n'authentifie pas explicitement l'expéditeur d'un message qui lui est destiné.

Problème 3 (9 points)

On considère une application dans laquelle un ensemble de clients utilisent un système de messagerie. Un client peut envoyer un message à un autre, à condition que les deux clients se soient enregistrés au préalable dans le système. Les opérations sont les suivantes :

- `enregistrer (nom_client, mot_de_passe)` : enregistre le couple (`nom_client, mot_de_passe`)
- `envoyer_message (nom_client, message)` : envoyer un message au client spécifié (le dépose dans sa boîte aux lettres)
- `messages (nom_client, mot_de_passe)` : renvoie le ou les messages reçus par le client depuis le dernier appel de cette méthode ; vide la boîte aux lettres (un nouvel appel renvoie une séquence vide si aucun message n'est arrivé entre temps)

Le nom d'un client est une chaîne de caractères de taille inférieure ou égale à 30, le mot de passe une chaîne de caractères de taille inférieure ou égale à 10. Un message est une structure comportant un en-tête et un corps. On supposera pour simplifier que l'en-tête comporte la date d'envoi (type prédéfini `date`) et le nom de l'expéditeur, et que le corps est une chaîne de caractère de taille ≤ 1000.

Question 1. On demande d'écrire en IDL CORBA l'interface utilisée par les clients de cette application. Indiquer avec précision toutes les exceptions devant à votre avis être prises en compte. Indiquer si une ou plusieurs opérations peuvent être *oneway*.

Question 2. On souhaite maintenant permettre à un client de se faire rappeler, s'il le souhaite, par le système de messagerie lorsque un nouveau message arrive dans sa boîte aux lettres (principe de l'appel en retour, ou *callback*), et d'exécuter alors (en parallèle avec le programme principal) une méthode `traiter_message(message)` d'un objet de classe `Rappel` que l'on spécifiera. On demande de compléter la réponse à la question 1 pour tenir compte de cette opération.

On rappelle les construction IDL pour les chaînes et séquences :

```
typedef string <[taille max]> type

typedef sequence <type, [taille max]> type
```

Question 3. Donner schématiquement en Java les programmes permettant de réaliser les fonctions définies à la question 2. On ne demande pas nécessairement tous les détails, mais on demande de donner la structure générale des programmes et d'expliciter les structures de données et méthodes qui permettent la réalisation de l'appel en retour.