Master2 Recherche - Systèmes et Logiciel. École Doctorale Math. Informatique Grenoble
Année 2003-2004

Construction d'Applications Réparties et Parallèles
Examen

16 mars 2004

Durée: 3 heures. Documents autorisés : notes personnelles, documents distribués en cours

---

*L'examen comporte 4 problèmes indépendants. Lire l'ensemble des énoncés avant de commencer à répondre. La longueur des énoncés **n'est pas** signe de difficulté mais elle est nécessaire à une bonne spécification des problème ; le fait qu'un énoncé soit long ne signifie pas que la réponse doive elle-même être longue (c'est même généralement le contraire !). La **clarté**, la **précision** et la **concision** des réponses, ainsi que leur **présentation matérielle**, seront des éléments importants d'appréciation.*

Prière de rédiger les réponses au problème 4 sur une feuille séparée.

Problème 1 (5 points)

Dans ce problème, on s'intéresse à un système de communication par événements de type *publish-subscribe*. On peut définir des sujets (que l'on notera par des identificateurs), et les opérations sont les suivantes :

subscribe(subject) - s'abonner au sujet *subject*.

unsubscribe(subject) - se désabonner du sujet *subject*.

publish(event, subject) - émettre un événement *event* sur le sujet *subject*. On peut associer une information quelconque (par ex. fichier) à l'événement *event*.

attach(handler, subject) – déclarer un programme traitant *handler* qui sera activé lors de la réception d'un événement sur le sujet *subject* ; on suppose que ce mécanisme d'activation permet au processus activé d'avoir accès à l'événement reçu.

Les sujets sont définis par un administrateur et on suppose que les utilisateurs peuvent consulter la liste des sujets courants. Un utilisateur peut s'abonner à plusieurs groupes.

La question 1) porte sur l'utilisation de ce système. La question 2 porte sur sa réalisation au moyen d'un système de communication de groupe. Les deux questions sont indépendantes. Les réponses à ces questions comporteront des programmes commentés pour lesquels on utilisera une notation algorithmique de « pseudo-code » (on n'impose pas de langage particulier). On aura besoin de faire exécuter des processus. Pour cela, on supposera disponibles les deux primitives suivantes :

run (prog, params) permet de créer et de lancer un nouveau processus avec le programme *prog* et de lui passer les paramètres *params*.

bind(prog, id) permet d'associer un nom (identificateur) *id* au programme *prog* ; le programme associé à un nom donné *id* est alors noté *prog*[*id*].

Question 1. On souhaite utiliser le système ci-dessus pour réaliser une application de surveillance d'une installation. L'installation comporte un ensemble de modules logiciels de surveillance associés à des organes matériels. Ces organes peuvent dynamiquement être ajoutés ou retirés. Lorsqu'un organe O_i est ajouté, il doit s'intégrer dans deux groupes spécifiés : un groupe S_i auquel il participe comme émetteur d'événements et un ensemble de groupes C_{i1}, C_{i2}, … , C_{in} auxquels il participe comme récepteur d'événements. En tant

qu'émetteur, il doit périodiquement diffuser son état (soit $etat_i$) au groupe S_i. En tant que récepteur, il doit exécuter un traitant $trait_i$ lorsqu'il reçoit un événement.

a) Expliquer avec précision (sous forme d'un programme commenté) les opérations à effectuer lors de l'ajout et du retrait d'un organe O_i.

b) Une manière systématique de réaliser ces opérations consiste à utiliser un fichier de configuration. Quelles informations doit contenir ce fichier, et comment ces informations sont-elles utilisées ? Quel est l'avantage de procéder ainsi ?

c) Le logiciel associé à un organe O_i comporte plusieurs processus. Expliquer avec précision pourquoi plusieurs processus sont nécessaires, à quel moment ils sont activés, et donner schématiquement leur programme. On suppose qu'on dispose d'un opérateur **every** T **do** A qui permet d'effectuer l'opération A toutes les T unités de temps.

Question 2. On suppose que l'on dispose d'un système de communication avec gestion de groupes, fournissant les primitives suivantes :

group-desc = create(group-id) créer un groupe identifié par l'identificateur *group-id* (un nom symbolique). Cette primitive renvoie un descripteur *group-desc* qui sera utilisé pour toutes les opérations ultérieures sur le groupe.

join (group-desc) l'utilisateur appelant rejoint le groupe désigné par *group-desc*. Un utilisateur peut être membre de plusieurs groupes

leave (group-desc) l'utilisateur appelant quitte le groupe désigné par *group-desc*.

send (message, group-desc) envoie le message *message* au groupe désigné par *group-desc*. Ce message sera délivré à tous les membres du groupe, comme indiqué ci-après.

r = wait() permet à un utilisateur membre d'un ou plusieurs groupes d'attendre un message. Le résultat *r* de cette opération désigne un couple (*r.message, r.group-id*) où *r.message* est le message reçu et *r.group-id* est l'identificateur du groupe où ce message été émis.

On demande de donner le programme de la réalisation des primitives *subscribe*, *unsubscribe*, *publish* et *attach* en utilisant les primitives de communication définies ci-dessus.

Problème 2 (5 points)

Ce problème on examine quelques aspects d'un système à composants. Un composant comporte un état et un ensemble de méthodes qui agissent sur cet état. Le système est mis en œuvre par une infrastructure de programmation utilisant des intercepteurs. Soit *C.meth(params)* une méthode d'un composant *C*, telle qu'elle a été développée par le programmeur de l'application. Lorsque le composant est exécuté dans l'infrastructure, l'appel de *meth* conduit à exécuter en fait la séquence

prélude (meth) ;
meth(params) ;
postlude (meth) ;

dans laquelle *prélude* et *postlude* sont contenus dans l'intercepteur. Par ailleurs une méthode

init(C) est exécutée au chargement de *C* (avant la première exécution d'une méthode de *C*).

Dans ce problème, on s'intéresse à l'utilisation des intercepteurs pour réaliser des propriétés particulières pour le composant.

Question 1. On souhaite que le composant ait la propriété de persistance. Cette propriété est obtenue en associant au composant *C* un fichier *f(C)* dans lequel est sauvegardé l'état de *C*. On suppose que le composant *C* fournit une méthode *sauvegarder()* qui copie dans *f(C)* la partie de l'état qui a été modifiée depuis la dernière sauvegarde et une méthode *restaurer()* qui restaure l'état de *C* à partir du fichier *f(C)*. On suppose par ailleurs que le composant *C* peut être passivé par le système.

a) Rappeler en quoi consiste l'opération de passivation et quelle est son utilité. Cette opération est-elle visible aux utilisateurs de *C* ?

b) Indiquer ce que doivent contenir les séquences *init*, *prélude*, et *postlude* pour assurer la persistance. Introduire si besoin les informations suplémentaires nécessaires.

Question 2. On souhaite contrôler l'utilisation du composant *C* au moyen de listes d'accès. Rappeler la définition d'une liste d'accès. Montrer que l'on peut définir des listes d'accès différentes pour chaque méthode de *C*. Indiquer dans les grandes lignes comment cela peut être réalisé : que faut-il mettre dans *init*, *prélude*, et *postlude* ?

Question 3. L'intercepteur d'un composant *C* peut-il être engendré automatiquement ? À partir de quelles informations ? indiquer le principe de cette génération.

Problème 3 (5 points)

Ce problème est consacré à l'examen de quelques opérations utilisant les techniques de la cryptographie. On rappelle les notations : si *M* est un message, on note $C=\{M\}_K$ le résultat du chiffrement de *M* en utilisant une clé *K* ; on note $M=[C]_K$ le résultat du déchiffrement du message chiffré *C* en utilisant une clé *K*.

Les 3 questions sont indépendantes.

Question 1. Soit deux correspondants A et B qui communiquent entre eux en utilisant un protocole de cryptographie à clé publique. On note respectivement KPA et KPB les clés publiques de A et B, KSA et KSB les clés privées correspondantes de A et B. On considère les 2 messages ci-après, envoyés par A à B (où M désigne un message « en clair » non spécifié) :

a) $\{\{M\}_{KSA}\}_{KPB}$

b) $\{\{M\}_{KPC}\}_{KPB}$ (C est un correspondant distinct de A et B et KPC est sa clé publique)

Pour chacun de ces messages, dites quelle est d'après vous sa fonction, quelles garanties éventuelles il procure, et expliquez comment il sera traité par B lors de sa réception.

Question 2. Pour réaliser un système de paiement électronique, on utilise des "chèques électroniques" signés par le titulaire du compte, avec un système à clés publiques. Un tel chèque comporte une information *M* ainsi qu'une "signature" $\{D(M)\}_{KSN}$. *D* est une fonction de hachage non inversible, *M* contient le nom *N* du titulaire du compte, le nom *B* de la banque et la somme *S*, et enfin *KSN* est la clé secrète de *N*. On suppose que la clé publique de

N, KPN, est accessible à tous. Une personne recevant un chèque en paiement l'envoie à la banque.

a) Expliquer comment la banque vérifie l'authenticité de la signature et l'intégrité du chèque (c'est-à-dire sa non-modification après signature).

b) Le schéma ci-dessus permet à une personne de faire une copie du chèque et de l'envoyer à la banque pour se faire payer deux fois. Proposez une modification simple pour éviter cela.

Question 3. On veut réaliser un système d'authentification en utilisant la cryptographie à clé secrète. Soit A et B deux partenaires qui veulent s'authentifier mutuellement, puis communiquer entre eux en utilisant le chiffrement à clé secrète. Ils font appel à un serveur d'authentification S qui connaît la clé secrète de A, KA et la clé secrète de B, KB. Le protocole se déroule comme suit :

$A \rightarrow B : I, A, B, \{RA, I, A, B\}_{KA}$ où I est un numéro d'ordre (incrémenté à chaque authentification) et RA un nombre choisi au hasard.

$B \rightarrow S : I, A, B, \{RA, I, A, B\}_{KA}, \{RB, I, A, B\}_{KB}$ où RB est un autre nombre tiré au hasard.

$S \rightarrow B : I, \{RA, K\}_{KA}, \{RB, K\}_{KB}$, où K est la clé secrète choisie par S pour la communication entre A et B.

$B \rightarrow A$: ???

a) Quel est le dernier message envoyé de B vers A, qui conclut le protocole d'authentification ?

b) Expliquer comment A et B peuvent chacun avoir l'assurance que son correspondant est bien celui qu'il prétend être. Expliquer le rôle de I, RA et RB.

Problème 4 (5 points) Notation : 3,5 pour la partie 1, 1,5 pour la partie 2

1. Introduction à la programmation d'applications parallèles

1.1 **HPF/OpenMP/MPI**. Trois approches autour d'HPF, OpenMP et MPI vous ont été présentées pour la construction d'application parallèles pour le calcul scientifique hautes performances. Présentez et commentez pour ces trois approches leurs avantages et leurs inconvénients.

1.2 **Méthodes et démarches.** Quelles méthodes et quelles démarches suivriez vous :

- Si vous aviez une application parallèle à réaliser de zéro avec pour objectif d'obtenir les meilleures performances et que vous disposiez de moyens importants (hommes, durée, financier) ?
- Si vous aviez à paralléliser une application séquentielle existante avec des moyens limités (hommes, durée, financier) ?

2. Outils d'exploitation et de gestion des grappes de grande taille

- Quels sont les problèmes soulevés par le passage à l'échelle dans ce contexte ?
- Quels sont les principales solutions qui peuvent être apportées ?