DEA ISC

Construction des Applications Réparties

(mars 1999)

Des réponses précises et concises sont demandées

L'objet du problème est l'étude d'une version répartie d'un *système de vente aux enchères* (une marchandise, proposée par un vendeur à plusieurs acquéreurs, est finalement vendue à celui qui a fait l'offre la plus intéressante).

Le vendeur est représenté par un processus *V* s'exécutant sur un site donné. Les acquéreurs sont représentés par des processus *Ai* qui s'exécutent sur des machines reliées par un réseau. La vente se déroule de la manière suivante :

1. Le vendeur transmet aux acquéreurs la description de la marchandise, avec une mise à prix initiale, et attend les propositions des acquéreurs. Le temps d'attente est borné par un "chien de garde" (*watchdog*). Les réponses qui ne sont pas parvenues dans le délai sont interprétées comme un abandon de l'acquéreur (il ne participe plus aux enchères).
2. Après visualisation sur son écran de la nature de la marchandise et du prix de référence, un acquéreur renvoie au vendeur une proposition de prix (supérieure ou égale au prix de référence).
3. Le vendeur choisit la proposition la plus intéressante, qui devient le nouveau prix de référence. Il transmet ce nouveau prix à l'ensemble des acquéreurs, et attend leurs réponses dans les mêmes conditions qu'à l'étape 1.
4. Un acquéreur visualise le nouveau prix proposé et renvoie, soit une nouvelle proposition, soit un signal d'abandon (dans ce cas il ne participe plus aux enchères).
5. Ces échanges se répètent jusqu'à ce qu'il n'y ait plus qu'une seule proposition en provenance des acquéreurs. Dans ce cas la vente est terminée.

Dans la suite, on étudie l'architecture de cette application selon différents modèles d'exécution présentés ci-dessous :

A - Modèle client-serveur.

Les processus clients sont les acquéreurs et le processus serveur est le vendeur.

Les primitives disponibles pour la programmation de l'application sont les suivantes :

- Coté client : *Call proc_name (serveur_Id, paramètres_appel, paramètres_retour)*

  Appel de la procédure *proc_name* du serveur *serveur_Id*. Si la procédure *proc_name* n'existe pas ou si le serveur *serveur_Id* n'existe pas une erreur est retournée.

Le système de RPC sous-jacent est chargé de localiser le serveur, d'acheminer la requête correspondant à l'appel et de ramener le résultat au processus client.

B - Modèle client-serveur (bis).

Le processus client est le vendeur et les processus serveurs sont les acquéreurs.

Les primitives sont les mêmes que dans le cas précédent.

C - Modèle à messages.

Les acquéreurs et le vendeur sont représentés par des processus qui communiquent par échange de messages. Chaque processus est désigné par un identificateur universel (noté *Proc_Id*) sur l'ensemble du système.

Les primitives disponibles pour la programmation de l'application sont les suivantes :

- *Send (Proc_Id, msg)*

  Envoi du message *msg* au processus *Proc-Id*. Si *Proc_Id* n'existe pas une erreur est retournée.

  Cette primitive n'est pas bloquante (appel asynchrone).

- *Receive (Proc_Id, msg)*

  Attente d'un message en provenance du processus *Proc_Id*. Le message reçu est stocké dans la variable *msg*. Cette primitive est bloquante, jusqu'à réception d'un message.

Le système sous-jacent est chargé de la localisation des processus et de l'acheminement des messages.

D - Modèle événement-réaction.

Comme dans le cas précédent, les acquéreurs et le vendeur sont représentés par des processus autonomes. La communication entre les processus est réalisée par émission d'événements auxquels sont associées des réactions.

Les primitives disponibles pour la programmation de l'application sont les suivantes :

- *Subscribe ("nom_événement", "réaction")*

  Cette primitive permet à un processus de s'abonner aux événements de type "*nom_événement*". Lorsqu'un événement de ce nom est émis par un processus, le programme exécutable de nom "*réaction*" est exécuté.

  Un abonnement est un triplet <*Proc_id, "nom_événement", "réaction"*>, où *Proc_id* est l'identificateur du processus utilisateur exécutant la primitive *Subscribe.*

  Il n'y a qu'un seul abonnement à un événement pour un processus donné ; cela veut dire que si un processus s'abonne plusieurs fois à un événement avec des réactions différentes, le dernier abonnement annule le précédent.

- *Unsubscribe ("nom_événement")*

  Cette primitive annule l'abonnement pour le processus donné.

- *Publish ("nom_événement", paramètres)*

  Cette primitive permet à un processus utilisateur d'émettre un événement de type "*nom_événement*". Cette primitive n'est pas bloquante pour le processus émetteur.

  Le système sous-jacent (bus logiciel) établit la relation entre l'émission d'un événement et l'exécution des réactions correspondantes. Toutes les réactions associées à cet événement, s'il y

en a, sont exécutées en parallèle. Un processus est créé pour chacune d'elle. Ce processus reçoit les paramètres associés à l'événement lors de son émission. Ce processus est détruit lorsque l'exécution de la réaction est terminée.

Un événement est un signal temporaire : il n'est pas mémorisé par le bus logiciel. En conséquence, s'il n'y a pas d'abonnement au moment où l'événement est émis, il n'y a aucune réaction exécutée et l'événement est perdu.

Question 1

On suppose que les acquéreurs connaissent l'identité du vendeur. En revanche le vendeur ne connaît pas l'identité des acquéreurs avant une vente. Dans certains cas cette connaissance est nécessaire à la réalisation de la vente. Pour les modèles de programmation où c'est nécessaire, décrire un protocole simple qui permette d'établir cette phase de reconnaissance initiale.

Question 2

Décrire pour chaque modèle de programmation l'algorithme du processus *V*.

Question 3

Décrire pour chaque modèle de programmation l'algorithme d'un processus acquéreur *Ai*.

Question 4

Pour chaque modèle de programmation, on s'intéresse à réaliser les fonctions suivantes :

- possibilité pour un acquéreur de se connecter en cours de vente (c'est à dire de participer aux enchères à tout instant).
- possibilité pour un vendeur de réaliser plusieurs ventes en parallèle (pour des produits différents).

Dans le cas où c'est possible, on décrira brièvement le principe de la solution retenue.

Question 5

Si vous aviez à construire une telle application, quelle plate-forme choisiriez-vous ? Justifier votre choix en quelques lignes (20 lignes maximum).