DEA d'Informatique - Systèmes et Communication
INPG-UJF Grenoble

Principes des Systèmes Répartis
Examen

9 décembre 1997
Durée: 3 heures, documents autorisés

---

*L'examen comporte 3 problèmes indépendants. La longueur des énoncés n'est pas signe de difficulté mais elle est nécessaire à une bonne spécification des 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).

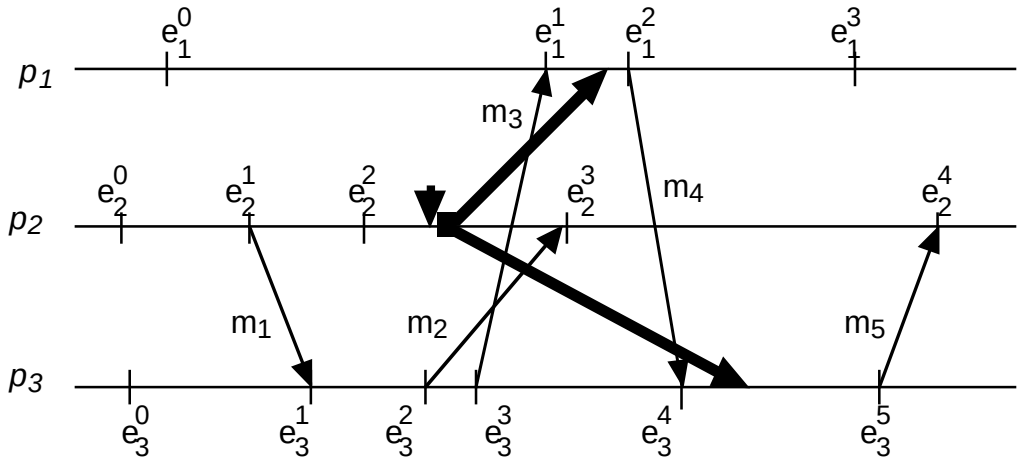
Les trois question sont indépendantes.

Question a). On considère un système de trois processus muni d'horloges vectorielles. On considère deux événements *e* et *e'* respectivement datés par (2, 1, 5) et (6, 3, 5).

- Quelle est la relation de précédence causale entre *e* et *e'* ? Construire (et représenter graphiquement) un scénario de déroulement du système conduisant à la production de *e* et *e'*.

- On modifie la date de *e'* en (6, 3, 4). Que devient la réponse aux deux questions ci-dessus ?.

Question b). On considère le système de 3 processus décrit ci-dessous. Le processus p_2 lance l'algorithme de Chandy-Lamport pour l'enregistrement d'état, au point indiqué sur la figure. La communication est FIFO entre deux sites.



On demande de compléter l'exécution de l'algorithme en indiquant les messages échangés. Quel est l'état enregistré par l'algorithme (état des sites et messages en transit) ?

Question c). On considère un système réparti composé de *n* processus $p_1, ... p_n$ dans lequel la durée de propagation des messages est bornée par une constante Δ, et dans lequel on dispose sur chaque site *i* d'une horloge donnant un temps réel approximatif T_i. La dérive entre deux sites quelconques est bornée par une constante δ (autrement dit, pour tout couple *i, j* on a $|T_i - T_j| < \delta$.

Expliquer comment, dans ces conditions, réaliser simplement l'algorithme de Chandy-Lamport.

Problème 2 (5 points)

On s'intéresse à la gestion de copies multiples d'une même donnée *x*. Ces copies (soit $x_1, ..., x_i, ... x_n$) sont maintenues respectivement par un ensemble de processus répartis (soit $p_1, ..., p_i, ... p_n$). On souhaite que le comportement du système soit "équivalent" à celui d'un système comportant un exemplaire unique de *x*.

Question a). Définir de manière précise cette équivalence, en utilisant par exemple la propriété de linéarisabilité. Donner des conditions suffisantes permettant d'assurer cette équivalence.

Question b). On suppose que le système est réalisé comme suit. Pour lire *x*, le processus p_i lit la copie locale x_i. Pour modifier *x*, le processus p_i met à jour x_i, puis diffuse à tous un message *(mise à jour, valeur)*. Lorsqu'un processus reçoit un tel message, il met à jour sa copie locale.

Montrer que cette méthode est incorrecte, même en l'absence de pannes. Indiquer le principe d'une solution correcte utilisant un mécanisme de diffusion. Quelle propriété doit avoir ce mécanisme de diffusion ?

Question c). On utilise maintenant une méthode différente, utilisant un jeton unique circulant sur un anneau virtuel reliant tous les processus (on suppose néanmoins que les communications entre processus, autres que la circulation du jeton, ne sont pas restreintes à cet anneau). La lecture fonctionne comme en a). Seul le processus p_i qui possède le jeton peut mettre à jour sa copie. Le protocole de mise à jour est le suivant : p_i met à jour x_i, puis diffuse à tous (par diffusion fiable) un message *(mise à jour, valeur)*, et enfin transmet le jeton à son successeur sur l'anneau. Un processus qui reçoit un message de mise à jour met à jour sa copie locale ; un processus qui reçoit le jeton exécute le protocole ci-dessus s'il a une mise à jour à faire ; sinon, il passe le jeton au suivant.

Montrer que le protocole ci-dessus est incorrect, et proposer une modification simple pour le corriger.

Question c). Avec le protocole (modifié) de la question b), décrire de manière précise le protocole mis en œuvre par un processus p_i pour se réinsérer après une panne.

Problème 3 (10 points)

On souhaite réaliser un protocole de gestion de groupes de processus. On utilise pour cela un séquenceur, qui est un processus particulier du groupe. Le séquenceur est chargé d'une part de réaliser la diffusion de messages aux membres du groupe, d'autre part de maintenir la liste courante des membres du groupe. Des processus peuvent quitter et rejoindre le groupe ; un processus peut tomber en panne (panne franche, *fail-stop*), et se réintégrer après réparation. On suppose que la communication est fiable et que le temps de transmission d'un message est borné par une constante connue *T*. La primitive de communication que l'on veut réaliser est une diffusion fiable totalement ordonnée : un message diffusé parvient à tous les processus corrects (non en panne), ou à aucun ; les messages sont reçus dans le même ordre par tous les processus.

Pour réaliser la gestion de groupes et la diffusion, on dispose d'un système de communication primitif qui fonctionne comme suit :

émission : la primitive *send(m, dest)* envoie le message *m* au processus destinataire désigné par *dest*.

réception : le processus destinataire est interrompu et exécute une séquence de programme "réception" (elle-même ininterruptible). Au début de cette séquence, deux variables *m* et *exp* sont mises à jour : *m* contient le message reçu, *exp* l'identité du processsus expéditeur. La suite de la

séquence contient le programme de traitement de la réception du message ; ce programme peut donc utiliser *m* et *exp*.

On utilisera une variable interne *seq* accessible à tous les processus, pour désigner à tout instant l'identité du séquenceur. On traitera les questions a) b) c) en supposant résolu le problème de l'initialisation du groupe, et sans prendre en compte le problème de la panne du séquenceur. Ces questions sont abordées en d) et e).

Question a). Expliquer comment le séquenceur peut maintenir la liste de processus présents. Donner l'algorithme des opérations qui permettent à un processus de rejoindre ou de quitter un groupe.

Question b). Donner l'algorithme qui permet de réaliser la diffuision fiable totalement ordonnée.

Question c). Même question que b), mais en supposant maintenant que le système de communication peut perdre des messages (mais toujours avec l'hypothèse du délai de transmission borné : soit un message arrive avec un délai inférieur à *T*, soit il n'arrive pas.

Question d). Proposer un mécanisme pour initialiser un groupe de processus.

Question e). Donner l'algorithme du traitement de la panne du séquenceur.

Question f). Dans cette question, on suppose maintenant que la communication est fiable et asynchrone (un messages parvient toujours à destination, mais le temps de transmission n'est pas borné). Indiquer quel détecteur de fautes serait nécessaire :

- pour pouvoir continuer à maintenir la liste des processus présents

- pour pouvoir assurer la diffusion fiable ordonnée.