DEA d'Informatique - Systèmes et Communication. Année 2002-2003
INPG-UJF Grenoble

Algorithmique et Techniques de Base des Systèmes Répartis
Examen

18 décembre 2002
Durée: 3 heures, documents autorisés

---

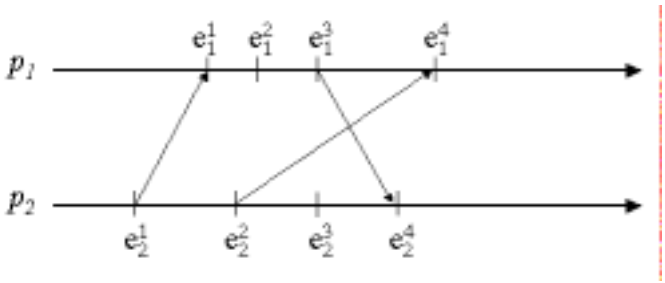
*L'examen comporte 3 problèmes indépendants. Lire l'ensemble des énoncés avant de commencer à répondre. La longueur des énoncés n'est pas signe de difficulté mais elle est nécessaire à une bonne spécification des problèmes. La **clarté**, la **précision** et la **concision** des réponses, ainsi que leur **présentation matérielle**, seront des éléments importants d'appréciation.*

Prière de rédiger les réponses aux 3 problèmes sur 3 feuilles séparées.

Problème 1 (7 points)

Les deux questions A et B sont indépendantes.

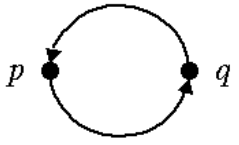
A. On considère le système décrit par le schéma ci-dessous.



a) Représenter graphiquement le treillis des états cohérents de ce système.

b) On considère les deux états (e_1^3, e_2^2) et (e_1^2, e_2^3). On suppose qu'une propriété P du système est vraie dans ces deux états. Que peut-on dire de P, du point de vue de l'ensemble des exécutions possibles du système ? Même question pour le couple d'états (e_1^2, e_2^1) et (e_1^1, e_2^2)?

B. On considère deux processus *p* et *q* qui communiquent en échangeant des messages sur un anneau unidirectionnel FIFO (schéma ci-dessous). Quand un processus reçoit un message, il fait un traitement, puis envoie un message à l'autre. Les processus fonctionnent en bascule : à un instant donné, il y a au plus un message en circulation dans le système. On caractérise l'état de chaque processus par le nombre total de messages qu'il a envoyés ; chaque processus passe donc par la suite d'états numérotés 0, 1, 2, etc. On suppose que *p* envoie le premier message.



a) On suppose que chaque processus enregistre son état immédiatement après l'envoi de chaque message. Cet enregistrement est-il sujet à l'"effet domino" ? Même question si chaque processus enregistre son état immédiatement après la réception de chaque message.

b) On suppose que p est dans l'état 51 et qu'il envoie alors un message à q. Immédiatement après cet envoi, il lance l'algorithme d'enregistrement d'état de Chandy et Lamport (on rappelle que cet algorithme enregistre un état global constitué de l'état des divers processus et de leurs canaux de communication). Décrire avec précision (utiliser un schéma) le déroulement de cet algorithme et indiquer le ou les résultat(s) possible(s).

c) On suppose maintenant que les canaux de communication ne sont plus FIFO. Montrer (par un exemple) que l'algorithme de Chandy et Lamport peut conduire à enregistrer un état incohérent. Proposer une modification de cet algorithme, adaptée au présent système, pour qu'il enregistre toujours un état cohérent.

Problème 2 (7 points)

La question d) dépend de la question c) ; il n'y a pas d'autre dépendance entre les questions.

On considère un système réparti dans lequel les processus peuvent avoir des défaillances de type « panne franche » (*fail stop*), sans reprise (la défaillance est définitive). Un processus est dit correct s'il n'est pas défaillant. On suppose que tout processus correct peut toujours communiquer avec tout autre processus correct (les défaillances ne partitionnent pas l'ensemble des processus corrects).

a) On suppose le système asynchrone. On veut réaliser un protocole de diffusion fiable *uniforme*,défini par les propriétés suivantes : si un processus correct diffuse un message *m*, ce processus délivre *m* ; si un processus (correct ou incorrect) délivre un message *m*, tous les processus corrects délivrent ce message *m* ; si un processus correct délivre *m*, alors il le délivre une seule fois, et *m* a été diffusé par un processus).

On propose l'algorithme suivant pour la diffusion fiable :

Exécution de *broadcast (m)* par processus *p* [diffusion de *m*]:

```
Marquer m avec (émetteur de m, numéro de séquence) // identifie m
Exécuter send(m) vers tous les voisins (y compris p)
```

Exécution de *deliver (m)* par processus *p* [livraison de *m*]:

```
Lors de la première exécution de receive (m) :
   Exécuter deliver (m) (au processus p), si p ≠ émetteur
   Exécuter send(m) vers tous les voisins
```

Les primitives *send* et *receive* sont les opérations élémentaires d'envoi et réception de message point à point, supposées fiables (un message envoyé parvient à destination si l'émetteur et le récepteur sont corrects). Les voisins d'un processus *p* sont les processus *q* tels qu'il existe un canal direct de communication de *p* vers *q*.

Montrer que l'algorithme ci-dessus est incorrect, en donnant un scénario qui viole les spécifications. Corriger l'algorithme. Quel est l'intérêt pratique de considérer un protocole de diffusion uniforme (c'est-à-dire prenant en la livraison de messages à tous les processus, corrects ou incorrects) ?

b) On suppose toujours le système asynchrone. On rappelle qu'un détecteur de défaillances construit et maintient à jour, pour chaque processus, une liste de processus considérés comme défaillants par ce processus. On rappelle que la classe *S* de détecteurs de défaillances est caractérisée par les propriétés (complétude forte, exactitude faible). On définit la classe *W* comme la classe ayant les propriétés (complétude faible, exactitude faible). En supposant disponibles un détecteur de défaillances de la classe *W* ainsi qu'une primitive de diffusion fiable *broadcast(m)*, telle que spécifiée à la question a) ci-dessus, expliquer comment construire un détecteur de la classe *S* et donner l'algorithme correspondant.

c) On suppose maintenant que le système est synchrone ; la durée de transmission d'un message est bornée par une constante connue Δ. Donner l'algorithme d'un détecteur de défaillances utilisant la technique du *heartbeat* (envoi périodique de messages « je suis vivant » par chaque processus). Quelle est la classe (*W, S,* …) de cet algorithme ?.

d) Modifier l'algorithme donné en réponse à c) pour l'adapter à un système asynchrone (indication : utiliser un délai de garde variable pour réduire les fausses détections de pannes). Quelle est la classe de cet algorithme ?

Problème 3 (6 points)

1 - Gestion d'une mémoire de stockage

a) Expliquer les principes directeurs du nommage relatif

b) Quels en sont les avantages ?

c) Comment se passe une migration d'objet ?

d) Quels problèmes pose ce schéma de nommage en mémoire d'exécution et comment ces problèmes sont ils résolus ?

2 - Implantation du modèle client-serveur

a) Quels sont les avantages d'un serveur multi-programmé (*multi-thread*) ?

b) On considère le cas d'un serveur (avec état modifiable) fiable dont les requêtes d'appel sont acheminée par un protocole non-fiable. Quel mécanisme permet de compenser une panne temporaire du réseau ?

c) Que faut il ajouter pour tolérer un redémarrage du serveur ?

3 – Implantation d'une mémoire partagée dupliquée

a) Expliquez la différence fondamentale entre *eager-release-consistency* et *lazy-release-consistency*.

b) En quoi consiste l'utilisation de *diffs* et quel est le bénéfice ?