Université Joseph Fourier
Master 2ème année Recherche – Systèmes et Logiciels - année 2004-2005

Algorithmique et techniques de base des systèmes répartis (SR)
Examen
16 décembre 2004

Durée : 3 heures (8h30-11h30) ; tous documents autorisés

N.B. L'examen comporte 4 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. Le fait qu'une question soit longue ne signifie pas que la réponse doit elle-même être longue (c'est même souvent le contraire). La **clarté**, la **précision** et la **concision** des réponses, ainsi que leur **présentation matérielle**, seront des éléments importants d'appréciation.

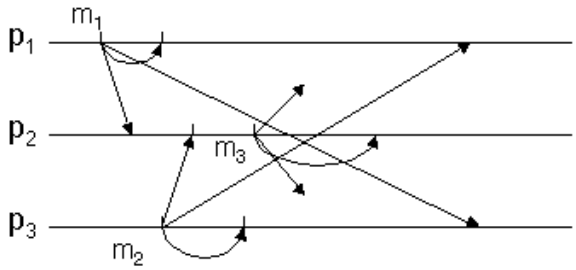
Problème 1 (5 points). Les questions sont indépendantes

Question 1. On considère un système d'horloges vectorielles $V_i[j]$, avec les règles de mise à jour indiquées dans le cours. Démontrer que sur toute coupure cohérente, pour tout i et tout j, $V_j[i] \leq V_i[i]$. Quelle est la propriété intuitive qui est traduite par cette inégalité ? Quelle est l'inégalité correspondante pour les horloges matricielles ?

Question 2. On réalise la diffusion causale au moyen d'horloges vectorielles « restreintes » comme indiqué dans le cours : les seuls événements pris en compte sont les diffusions de messages. Compléter le schéma ci-dessous en indiquant :

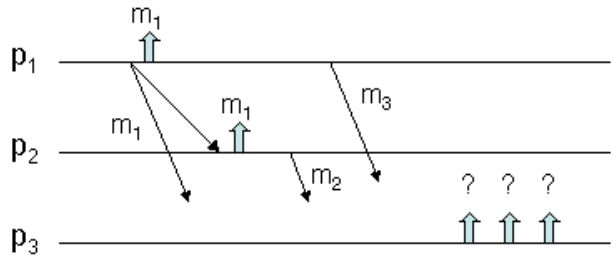
a) les moments où le message m_3 est délivré à p_1 et p_3

b) les valeurs des horloges vectorielles aux moment de la délivrance de tous les messages



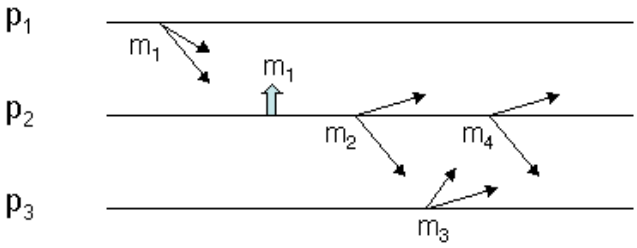
Question 3. Dessiner le schéma temporel de deux processus communiquant par messages, et tels que toute coupure cohérente (sauf l'état initial) comporte au moins un message en transit (c'est-à-dire qui traverse la coupure). Un schéma temporel est un dessin indiquant les événements et les messages, comme les schémas ci-dessus et ceux du cours. Le schéma le plus simple est le meilleur. Indications : utiliser une structure régulière (répétitive) pour chaque processus.

Question 4. On considère le système représenté ci-dessous, utilisant à la fois la communication point à point et la diffusion de messages. Les flèches verticales indiquent la délivrance des messages.



On demande de remplacer les ? par des numéros de messages (m_1, m_2, m_3) de manière que la délivrance des messages soit FIFO, mais non causale.

Question 5. On considère le système suivant, avec diffusion de messages (même notation pour la délivrance).



Indiquer toutes les séquences possibles de délivrance de tous les messages qui respectent l'ordre total mais non l'ordre causal.

Problème 2 (5 points)

On considère un système (qui peut être synchrone ou asynchrone), dans lequel la communication (primitives `send – receive`) est fiable (pas de perte de message) et les processus peuvent avoir des pannes franches. Soit l'algorithme de diffusion suivant réalisant une diffusion appelée `simple_broadcast` :

Pour diffuser un message `m`, un processus `p` exécute :

```
simple_broadcast (m):
  m.sender := #p ; m.seq := #seq(m)
  for all q ∈ voisins(p) do send(m, q)
  deliver(m)
```

Chaque processus exécute :

```
upon receive(m) do
deliver(m)
```

`#p` est le numéro du processus `p`, `#seq(m)` est un numéro de séquence interne à `p` et incrémenté de 1 à chaque diffusion de message (ainsi chaque message `m` a une identification unique donnée par `m.sender` et `m.seq`). Les voisins d'un processus `p` sont les processus `q` tels qu'il existe un lien direct entre `p` et `q`.

Question 1. Quelles propriétés possède cet algorithme de diffusion ? À quelle condition cet algorithme réalise-t-il une diffusion fiable ?

Question 2. Un message est dit k-stable s'il a été reçu par k processus. On demande d'étendre le protocole ci-dessus pour qu'un message ne soit délivré aux processus qui le reçoivent, et à son émetteur, que s'il est k-stable (on rappelle qu'il y a une différence entre **recevoir** et **délivrer**).

Problème 3 (8 points) Les questions sont indépendantes (sauf 3 qui dépend en partie de 1 et 2)

Question 1. On considère un système synchrone, avec communication fiable. Les processus peuvent avoir des pannes franches sans réparation. Le temps de transmission d'un message est borné par une constante connue δ. Les horloges des différents processus sont mutuellement synchronisées, de sorte que la valeur absolue de l'écart entre les horloges de deux processus soit inférieure à une constante connue *emax*.

On demande d'écrire le programme d'un détecteur de pannes de classe P utilisant la technique du *heartbeat*. On considère comme négligeable le temps de calcul sur les processeurs.

Question 2. On considère un système asynchrone avec pannes franches sans réparation, comportant n processus p_1, p_2, …, p_n). On suppose que l'on dispose d'un détecteur de pannes de classe P.

On rappelle qu'un algorithme d'élection choisit un processus élu unique (correct) et informe les autres processus de l'identité du processus élu. On souhaite réaliser un programme pour l'élection, avec les propriétés suivantes :

Sûreté. Si un processus p_i est élu à un certain instant, ou bien i=n, ou bien tous les processus p_k tels que $k > i$ sont en panne.

Vivacité. Si un processus correct lance l'élection, un processus sera élu au bout d'un temps fini.

a) Montrer que la propriété de sûreté implique l'unicité du processus élu.

b) Écrire un programme réalisant l'élection avec les spécifications ci-dessus, en utilisant le détecteur de classe P. Montrer que le programme répond aux spécifications.

Question 3.

Comment peut-on à votre avis modifier la définition du détecteur de classe P pour prendre en compte les pannes franches avec réparation ? Que deviennent alors les solutions des questions 1 et 2 ?

Question 4.

Une installation informatique se compose d'un serveur dupliqué (2 exemplaires). Chaque serveur a un MTTF de 5 jours et un MTTR de 6 heures. Les défaillances des deux serveurs sont indépendantes.

Quelle est la disponibilité globale du système (donner la valeur numérique ; vous pouvez faire le calcul à la main) ? Votre réponse dépend-elle de la technique utilisée (serveur primaire ou duplication active) ?

Problème 4 (2 points) Les questions sont indépendantes

Question 1. On considère la technique du vote pondéré pour la gestion de N copies multiples, avec un quorum en lecture *nr* et un quorum en écriture *nw*, tels que $nr + nw > N$.

a) Expliquer le but de cette technique.

b) À quelles situations correspondent respectivement les cas $nr = 1$, $nw = N$ et $nr = N$, $nw = 1$

Question 2. On considère l'exécution suivante de deux processus p_1 et p_2.

p_1 : R(x)1 ; R(x) 2 ; W(y) 1

p_2 : W(x)1 ; R(y) 1 ; W(x) 2

On suppose que x et y sont initialement égales à 0.

L'exécution ci-dessus est-elle séquentiellement cohérente ?

Question 3. Montrer que l'exécution ci-dessous n'est pas causalement cohérente.

p_1 : W(x)0 ; W(x) 1

p_2 : R(x)1 ; W(y) 2

p_3 : R(y)2 ; R(x) 0

Montrer qu'on peut la rendre causalement cohérente en permutant deux instructions.