Examen en Algorithmique et systèmes répartis

3-ième Année IMA et Master IT (Durée : 1H45)
Documents autorisés : documents distribués et notes de cours

Décembre 2008

1 Causalité et horloges de Mattern (10 points)

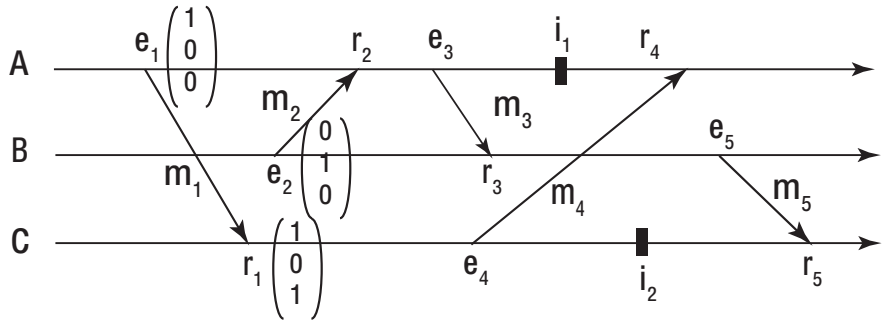

FIG. 1 – Diagramme événementiel d'une exécution répartie

Le diagramme de la figure 1 représente le début d'une exécution répartie. Les premiers événements sont datés à l'aide du mécanisme d'horloge de Mattern qui associe à tout événement x un vecteur horloge H_x. On note $x \prec y$ la relation de précédence causale et $H_x < H_y$ la relation d'ordre entre vecteurs horloges.

Questions (2 points par question)

1. Donner tous les chemins causaux débutant par les événements origines e_1 ou e_2 et aboutissant à l'événement r_4 dans le diagramme d'exécution de la figure 1. Même question pour les chemins causaux débutant par les événements origines e_1 ou e_2 mais aboutissant à l'événement r_5.
2. On suppose que l'exécution réelle a ordonné les événements dans le temps global réel selon la séquence suivante :

   $$e_1; e_2; r_1; r_2; e_3; i_1; r_3; e_4; r_4; e_5; i_2; r_5$$

   Préciser, en le justifiant, si cette séquence est **causalement** équivalente à celle de la figure 1. Autrement dit, représente-t-elle une exécution réelle équivalente à celle présentée dans la figure 1 ?
3. Compléter le diagramme en affectant leur vecteur horloge (leur date) aux événements non datés.
4. Déduire du calcul précédent si les événements internes i_1 et i_2 sont causalement liés ou pas.
5. Montrer que, pour deux événements x de vecteur horloge H_x et y de vecteur horloge H_y, si les composantes des deux vecteurs horloges pour un site s vérifient $H_x[s] < H_y[s]$, alors tout chemin causal aboutissant à x est passé pour la dernière fois sur le site s avant tout chemin causal aboutissant à y.

2 Interblocage et terminaison dans un calcul réparti (11 points)

On considère un calcul réparti composé de N processus (N fixé et connu) qui communiquent par messages via des canaux unidirectionnels. Le graphe orienté de communication est connu et fixe comme dans l'exemple de la figure 2 comportant 4 processus et 5 canaux. on suppose que tous les processus sont « accessibles » à partir du processus initial ($\exists$ au moins un chemin entre le processus initial P_0 et tout autre processus).

Les règles du calcul réparti constitué par ces N processus sont les suivantes :

- Tout processus reçoit des messages sur les canaux d'entrée ;
- Chaque processus P_i doit recevoir un message avant d'exécuter une étape ;
- Tout processus P_i, après l'exécution d'une étape envoie systématiquement un message sur chaque canal de sortie ;
- Au moins un processus « puits » n'a pas de canaux en sortie (P_3 dans la figure 2) ;

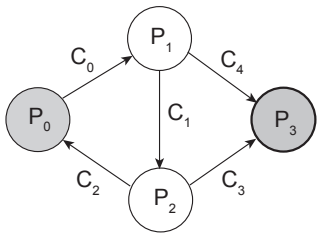

FIG. 2 – Graphe de communication

- Un processus initial (P_0 dans fig. 2) émet un message sur ses canaux de sortie pour activer le calcul.

En résumé, le comportement de tout processus participant obéit au schéma générique suivant[1] :

```
process P(i:0..N-1) {
  if (i==0) Emettre un message sur chaque canal de sortie ;
  while (true) {
    Recevoir un message sur l'un des canaux d'entrée ;
    Etape de calcul exploitant le message lu ;
    [ Emettre systématiquement un message sur chaque canal de sortie existant. ]
  }
}
```

Questions (1 point par question)

6. Dans quel modèle générique de calcul réparti peut-on classer le calcul décrit précédemment ?
7. Montrer la propriété stable suivante : si un processus appartenant à un cycle de communication (comme, par exemple, dans la figure 2, le cycle entre P_0, P_1 et P_2) reçoit un message, alors il existera désormais toujours un message qui circulera sur ce cycle.
8. Montrer que, pour tout cycle de communication, un processus au moins du cycle recevra finalement un message. En déduire que les processus ne peuvent pas être interbloqués même s'il existe des cycles dans le graphe de communication.
9. Montrer, par contre, que le calcul ne peut pas se terminer s'il existe au moins un cycle.
10. Donner une condition nécessaire et suffisante du graphe de communication pour que le calcul se termine en justifiant votre proposition.

Questions (2 points par question)

11. En supposant qu'il existe un **SEUL** processus puits, proposer un algorithme de terminaison permettant à ce processus puits de détecter la terminaison globale du calcul (s'il se termine !).
12. Adapter votre solution au cas d'un calcul comportant plusieurs processus puits.
13. On considère maintenant que chaque processus attend d'avoir reçu un message de **chaque** canal d'entrée avant de débuter une étape de calcul. Montrer que, dans un graphe sans cycle, le calcul est sans interblocage et se termine. Dans un graphe n'ayant qu'un puits, donner le critère très simple de détection de la terminaison en le justifiant.

[1] Un processus puits ne peut évidemment pas émettre de message puisqu'il n'a pas de canal de sortie.