Université Paris Diderot
M2 IAP
Algorithmes Réparties

Examen

Lundi 22 mars 2015

Durée 2 heures. 2 pages

Aucun document n'est autorisé. Tous les algorithmes considérés sont déterministes.

Exercice 1.— On considère un ensemble de n processus V sur un réseau de communication connexe $G = (V, E)$. Les liens de communication sont bidirectionnels. On définit $Voisin(p) = \{x \text{ tel que } (x, p) \in E\}$. Un processus Q est particularisé. On rappelle l'algorithme d'Echo (algorithme centralisé de vague) dans la figure 1.

Variables d'un processus x:
  rec_x: integer init 0
  $pere_x : V \cup \{udef\}$ init $udef$

CODE DE L'INITIATEUR Q:
  begin
    for all $x \in Voisin(Q)$ do send <tok> to x
    while $rec_Q < card(Voisin(Q))$ do
      begin receive <tok>; $rec_Q := rec_Q + 1$ end
    decide
  end

CODE DES NON INITIATEURS p:
  begin
    receive <tok> from x ; $pere_p := x$; $rec_p := rec_p + 1$
    for all $x \in Voisin(p)$ and $x \neq pere_p$ do send <tok> to x
    while $rec_p < card(Voisin(p))$ do
      begin receive <tok>; $rec_p := rec_p + 1$ end
    send <tok> to $pere_p$
  end

Figure 1: Algorithme d'ECHO.

1. Dans une vague produite par cet algorithme, combien de processus décident? 1 : Q
2. Dans une vague produite par cet algorithme, combien de messages sont échangés? $2(n-1)$
3. Le processus Q veut diffuser (broadcast) des messages à tous les processus. Décrire un algorithme qui après une phase préliminaire, utilise $n - 1$ communications (send/receive) quand Q veut diffuser un message (une description de l'algorithme est attendue, il est inutile d'écrire le code).

Exercice 2.— On considère un ensemble de n processus V sur un réseau de communication connexe $G = (V, E)$. Les liens de communication sont bidirectionnels. On définit $Voisin(p) = \{x \text{ tel que } (x, p) \in E\}$

Les processus ne connaissent pas G mais chaque processus p connait $Voisin(p)$. Un processus Q est particularisé.

Décrire un algorithme distribué qui permet à Q de connaitre G (une description de l'algorithme est attendue, il est inutile d'écrire le code).

Exercice 3.— On suppose que les processus sont organisés en anneau: un processus quelconque p peut communiquer avec son prédécesseur et son successeur. La taille de l'anneau, n, est *connue* de tous les processus. Les processus ne connaissent pas leur identité. On suppose qu'à chaque top d'horloge *exactement un* processus est activé et de façon atomique effectue un pas de calcul où il peut lire les variables de son prédécesseur et de son successeur et modifier ses propres valeurs. On suppose aussi que tous les processus sont activés infiniment souvent.

Est-il possible de faire un algorithme d'élection déterministe si $n = 5$. Si oui donnez un algorithme, sinon faites une preuve d'impossibilité.

Exercice 4.— On considère un système synchrone (les processus et les communications sont synchrones) de n processus (le graphe de communications est complet).

Répondez aux questions suivantes en justifiant vos réponses soit en citant un résultat (théorème, algorithme ...) du cours, soit par une preuve ou les grandes lignes d'un algorithme.

1. La communication n'est pas fiable (les messages peuvent être perdues). On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme de consensus qui tolère une panne par arrêt?
2. La communication est fiable. On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme de consensus qui tolère $\lceil n/2 \rceil$ pannes par arrêt?
3. La communication est fiable. On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme de consensus qui tolère $n - 1$ pannes par arrêt?
4. La communication est fiable. On considère des pannes de processus byzantines. Est-il possible de faire un algorithme de consensus qui tolère $\lceil n/2 \rceil$ pannes byzantines?

Exercice 5.— On considère un système asynchrone (les processus et les communications sont asynchrones) de n processus (le graphe de communications est complet).

Répondez aux questions suivantes en justifiant vos réponses soit en citant un résultat (théorème, algorithme ...) du cours, soit par une preuve ou les grandes lignes d'un algorithme.

1. On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme de consensus qui tolère une panne par arrêt?
2. On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme probabiliste de consensus qui tolère une panne par arrêt?
3. La communication est fiable. On considère des pannes de processus par arrêt. Est-il possible de faire un algorithme probabiliste de consensus qui tolère $n - 1$ pannes par arrêt?