Université Paris 7-Denis Diderot
Master 2 :
Algorithmique répartie
2013-2014

Devoir 3 :
à rendre pour le 17/03/2014 mars.

Dans ce devoir on suppose toujours 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, mais les processus ne connaissent pas leur identité. On rappelle que dans un algorithme d'élection tous les processus exécutent le *même* code.

1. Dans l'algorithme qui suit $Candidat_p$ est un booléen qui indique si p est candidat ou non. On suppose qu'initialement au moins un processus est candidat. $tirer()$ est une fonction de tirage aléatoire qui retourne $PILE$ ou $FACE$. On suppose que la probabilité de retourner $PILE$ est égale à la probabilité de retourner $FACE$: $Pr(PILE) = Pr(FACE) = \frac{1}{2}$. On suppose de plus que tous les tirages sont indépendants les uns des autres. Cet algorithme fonctionne en *rondes* dont la valeur est l'entier r (les échanges de messages entre processus ont lieu dans la même ronde pour tous)

```
for r := 0 to ∞ do
   if Candidat_p
   then
      V_p := tirer()
      send V_p to  Successeur_p
      receive  W from  Predecesseur_p
      if V_p = Pile ∧ W = Face then Candidat_p := False
   else
      receive W from Predecesseur_p
      send W to Successeur_p
   endif
```

FIGURE 1 – Algorithme non-déterministe.

   (a) Montrer que si, au début d'une ronde r, il y a au moins un candidat, à la fin de la ronde r il y aura aussi au moins un candidat.
   (b) En supposant qu'au début de la ronde r il y a au moins deux candidats, si p est candidat au début de la ronde r quelle est la probabilité qu'à la fin de la ronde r p ne soit plus candidat ?
   (c) En déduire que la probabilité qu'à la ronde r il y ait plus d'un candidat tend vers 0 quand r tend vers l'infini.
   (d) En utilisant cet algorithme transformer l'algorithme de la Figure 1 de façon à ce qu'il termine s'il n'y a plus qu'un seul candidat. En déduire que cet algorithme vérifie les propriétés suivantes : (1) s'il termine il ne reste plus qu'un seul processus candidat (2) la probabilité que cet algorithme termine est 1 (plus exactement la probabilité qu'il ne termine pas avant la ronde r tend vers 0 quand r tend vers l'infini).
   (e) Donner un algorithme qui, s'il n'y a qu'un seul candidat, donnera des identités différentes à chaque processus.

2. Dans cette partie, on modifie la façon dont les processus sont exécutés et échangent des informations. Comme précédemment les processus ne connaissent pas leur identité mais connaissent la taille de l'anneau.

   On suppose qu'à chaque top d'horloge exactement un processus est activé et de façon atomique (sans être interrompu) 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. [1]

   (a) Proposer un algorithme d'élection pour $n = 3$ et $n = 5$. (Avant de répondre aux questions suivantes il est conseillé d'étudier au préalable le cas $n = 4$)

   (b) On va montrer qu'il n'y a pas d'algorithme (déterministe) d'élection si n n'est pas un nombre premier. Pour cela, soit une configuration globale γ (une configuration globale est simplement l'état de chaque processus), on dira qu'un ensemble L est *symétrique* pour γ si (1) tous les $p \in L$ sont dans le même état pour γ (2) tous les prédécesseurs des éléments de L sont dans le même état pour γ (3) tous les successeurs des éléments de L sont dans le même état pour γ et (4) si p et q appartiennent à L alors p et q ne sont pas voisins (p n'est ni prédécesseur ni successeur de q).

      i. Montrer que si L est symétrique pour γ alors il existe une configuration γ' accessible à partir de γ telle que (1) L est symétrique pour γ' (2) si $p \notin L$ alors p est dans le même état en γ et γ' (3) chaque élément de L a fait au moins un pas de calcul.

      ii. En déduire que si l'ensemble des processus peut être partitionné en sous-ensembles $L_1, \cdots, L_k$ symétriques pour γ, il existe une séquence de calculs possible à partir de γ telle que tous les processus ont été activés infiniment souvent et telle qu'infiniment souvent $L_1, \cdots, L_k$ est une partition en sous-ensembles symétriques.

      iii. En déduire que s'il existe une partition $L_1, \cdots, L_k$ de sous-ensembles symétriques pour γ telle que tous les L_i contiennent au moins deux éléments il n'existe pas d'algorithme d'élection à partir de γ.

      iv. Montrant que si n n'est pas premier alors il n'existe pas d'algorithme d'élection (on rappelle que pour un algorithme d'élection tous les processus commencent dans le même état).

1. On suppose ici qu'un processus qui a terminé continue à faire indéfiniment des pas de calculs où il ne change pas d'état.