Université de Paris

UFR Informatique M2 IMPAIR - Année 2020–2021

Devoir Algorithmique Répartie n° 2

Devoir 2

Ce devoir est à rendre pour le 24 mars sur Moodle. Comme pour le précédent devoir vous pouvez le rendre par groupe d'au plus 3 étudiants à la condition que ce soit le résultat d'un travail de vraie collaboration entre tous les auteurs.

Exercice 1.— On considère une graphe connexe de communication avec des liens bidirectionnels avec des processus anonymes, mais chaque processus peut distinguer entre ses voisins (il peut distinguer ses liens de communication entrants et sortants). On suppose qu'il y a parmi les processus un *leader* unique : chaque processus a une variable `leader` cette variable est initialisée à False pour tous les processus sauf pour le leader pour lequel cette variable est égale à True.

1. Proposer un algorithme qui calcule le nombre de processus dans le graphe. (Dans un premier temps, le résultat de ce calcul pourra être connu du leader seul, dans un deuxième temps tous les processus devront avoir ce résultat)
2. Donner un algorithme permettant de donner un nom unique pris parmi les entiers entre 1 et N à chaque processus (chaque processus devra connaître son nom).
3. S'il n'y a pas de variable leader permettant de distinguer un leader, est-il possible d'avoir un algorithme déterministe permettant de nommer les processus (éventuellement avec un ensemble de nom plus grand que les entiers de 1 à N).

(Indication le premier algorithme pourra être une phase préliminaire du deuxième, on pourra utiliser l'algorithme de prob/echo. Pour la dernière question on pourra utiliser les résultats du cours)

Exercice 2.— 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, 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 de la Figure 1, $Candidat_p$ 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)
   (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 ?

```
1  for r := 0 to ∞ do
2     if Candidat_p
3     then
4        V_p := tirer()
5        send V_p to  Successeur_p
6        receive  W from  Predecesseur_p
7        if V_p = Pile ∧ W = Face then Candidat_p := False
8     else
9        receive W from Predecesseur_p
10       send W to Successeur_p
11    endif
```

FIGURE 1 – Algorithme non-déterministe d'election.

(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) Donner un algorithme (déterministe) qui teste s'il y a exactement un seul candidat sur l'anneau. Utiliser cet algorithme pour 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).