Université Jopseph Fourier – UFR-IMA
MST ESI2 : Examen d'algorithmique distribuée
Durée 2h

Tous documents autorisés – Le barême est indicatif

11 Avril 2000

Election par extinction

Il existe une forme particulière d'exclusion mutuelle qui consiste à ne faire exécuter une action donnée qu'une seule fois même si plusieurs processus le demandent. Une façon de garantir qu'à un instant donné de l'exécution d'un système distribué, seul un site effectue l'action est d'utiliser un algorithme d'élection qui décide du site devant exécuter l'action. Le principe de l'algorithme distribué d'élection par extinction est le suivant :

1. dès qu'un site désire effectuer l'action critique, il lance une élection. Pour cela, il initie une vague pour envoyer aux autres sites un message de candidature portant le nombre de *voix* dont il dispose,
2. si un site candidat reçoit un message de candidature d'un autre site, il **doit** participer à l'élection s'il ne l'a pas déja fait. Il ignore la candidature reçue si le nombre de voix proposé est inférieur aux siennes et il initie sa propre vague de candidature éventuellement. Sinon il accepte la candidature de l'autre site et propage la vague correspondante. [1].
3. Le site gagnant effectue l'action critique. Le gagnant est celui dont la vague n'est pas éliminée.

I - Election unique

Dans un premier temps, on considère qu'une **seule** exécution de l'action critique est à faire dans le système distribué. Naturellement, la façon de conduire l'élection dépend fortement de la géométrie du réseau que l'on considèrera fiable par la suite.

Election sur anneau (2 pts) Compléter le squelette de l'automate de contrôle exécuté sur chaque site du réseau pour faire une élection sur anneau monodirectionnel.

```
boolean élection = false; .....
repeter alt
   | besoin_d'élection  && !élection  ==>élection=true;
   |                  ...............;
   |                  lout!<CAND,ICI,.....>
   | lin?<CAND,src,......> && src!=ICI && élection  ==> .....
   | lin?<CAND,src,......> && dest!=ICI && !élection ==> ......
   | lin?<CAND,src,......> && src==ICI  ==> ......
fin_alt
```

La procédure `int Voix()` permet à un site d'obtenir son quota de voix. L'action critique est appelée `ACTION()`. La variable `ICI` donne l'identité du site l'exécutant. Le booléen `besoin_d'élection` local à chaque site ne devient vrai qu'une seule fois par site mais éventuellement sur plusieurs sites du système.

[1] On utilise les No des sites en cas de ballottage

Election sur clique **(2 pts)** Dans ce cas, le réseau est complètement maillé. Proposer un algorithme d'élection basé sur la vague dans une clique. Le message de retour de vague est marqué `ACK`. Compléter le squelette suivant:

```
boolean élection = false; .....
repeter alt  Quelque soit l de L
  |  besoin_d'élection && !élection  ==>élection=true;
  |                                        ...............;
  | l in L  l?<CAND,src......> && src==ICI  ==> .........
  | l in L  l?<CAND,src......> && src!=ICI && élection  ==> .....
  | l in L  l?<CAND,src......> && dest!=ICI && !élection ==> ......
  | l in L  l?<ACK,src......>  && .....................==> ......
  |                 .........
fin_alt
```

Election sur graphe bidirectionnel quelconque **(4 pts)** Proposer un algorithme d'élection basé sur des vagues de type *echo* explorant de façon récursive le graphe. On notera que, sur un site, on a seulement besoin du souvenir de la vague portant la dernière candidature gagnante.

II - Elections multiples

Le régime de fonctionnement du système est tel que des exécutions successives de l'action critique sont nécessaires. Le booléen `besoin_d'élection` local à chaque site passe de vrai à faux (et réciproquement) un nombre quelconque de fois dans le système. Il faut donc être capable de lancer des élections successives.

Séquentialisation implicite Dans un premier temps, on suppose que `besoin_d'élection` ne redevient vrai que postérieurement à l'exécution de l'action résultant de l'élection précédente. Il s'agit donc pour un site perdant de distinguer les candidatures d'élections différentes. Un mécanisme d'horloge logique d'élection parait approprié à ce type de problème :

1. Peut on borner cette horloge modulo X ? Que vaut X au moins ?**(2 pts)**
2. Modifier les trois algorithmes précédents pour permettre de multiples élections **(1+1+2 pts)**

Séquentialisation explicite Ici, on suppose que `besoin_d'élection` peut redevenir vrai n'importe quand. Il faut donc éviter de lancer une nouvelle élection avant la fin de l'exécution de l'action de l'élection précédente.

1. Comment synchroniser le lancement d'une élection avec la fin de l'action de l'élection précédente. Donner le principe d'une solution s'appliquant aux réseaux précédents.**(1 pt)**
2. Proposer une solution pour graphe bidirectionnel quelconque.**(3 pts)**

III - Liaison non FIFO

- La sérialisation des élections est elle toujours possible ? pourquoi ? **(1 pt)**
- Les algorithmes précédents sont ils corrects ? pourquoi ?**(1 pt)**