Algorithmique Distribuée
Election de Leader sur un Anneau Asynchrone

Janna Burman

janna.burman@lisn.fr

Hypothèses

- **Topologie** (graphe de communication) en forme **d'anneau**, bidirectionnel par défaut (si unidirectionnel, il est orienté)
- Par défaut, **chaque nœud a un identifient** (id) unique de l'ensemble UID $\subseteq \mathbb{N}^+$, |UID| >> n
- La taille n ne pas connue aux processus (par défaut)
- Les liens de communication sont FIFO
- **L'anneau est orienté** :
  - tout les n nœuds sont numérotés $p_0, p_1, \ldots, p_{n-1}$
  - ils apparaissent dans l'ordre correspondant à l'orienattion sur l'anneau
  - chaque nœud (processus) p_i sais quel lien sortant est vers le processus $p_{i+1 \bmod n}$ et lequel vers $p_{i-1 \bmod n}$ (vers p_{n-1} pour p_0)

Motivation pour Élection de Leader (EL)

- Simplifie la coordination (des nœuds)
- Problème important de brisure de la symétrie
  - utile pour nommage, comptage, etc.
  - peut aider de briser la symétrie dans le cas de deadlock (en supprimant l'entité élu de la cycle du deadlock)
- Aide à tolérer des défaillances (e.g., consensus Byz. avec leader)
- Peut être utile pour économiser les ressources (e.g. en aidant construire un arbre distribué de diffusion…).

Élection de Leader (EL)
la spécification du problème

Supposons que chaque nœud a une variable nommée *status* (ou *état)* t.q. status ∈ {leader, inconnu, non_leader}

1. A la terminaison (ou à partir d'une configuration), exactement un processus a status = leader (pour toujours)
2. Durant toute exécution, au plus un seul processus a status = leader

Les variantes :

A la terminaison (ou à partir d'une configuration), pour tout autre processus :

1. il apprend de ne jamais devenir leader (status = non_leader)
2. il connaît l'id du leader
3. il détermine que le leader est déjà élu
4. le leader termine explicitement

Algorithme de LeLann, Chang et Roberts (LCR)

pour un processus p_i

mon_id : UID ($\subseteq N^+$ - entiers positives) ;
mon_id := l'identificateur de p_i

état : {inconnu, leader, non-leader}; état := inconnu

lors de réveil spontané →
1. envoyer (mon_id) à p_{i+1}

lors de réception de message j →
2. si (réveil provoqué) alors envoyer (mon_id) à p_{i+1}
3. si (mon_id = j) alors
4. état := leader
5. sinon si (j > mon_id) alors
6. envoyer (j) à p_{i+1}

fin si

Preuve (LCR de base) - 1

Notons :

- $\mathbf{u_{max}}$ soit le processus avec id maximum
- $\mathbf{id_{max}}$ soit son id (id_i soit un id de processus p_i)

Lemme 1 : Dans le cas de réveil non-simultané, u_{max} est réveillé au plus après n-1 unités de temps (et envoie id_{max}).

Lemme 2 : u_{max} exécute ligne 4 et devient leader pour toujours.

Preuve : A partir du réveil de u_{max} prouvé par Lemme 1, tout processus, sauf u_{max} lui-même (ligne 3), qui reçoit $id_{max,}$ le relaye (ligne 6). Après des tels n-1 envois, u_{max} reçoit son id_{max} et devient leader. Il n'y a pas d'autres changements de variable *état* dans l'algorithme.

Preuve (LCR de base) - 2

Lemme 3 : Tout processus $p_i \neq u_{max}$ (avec $id_i < id_{max}$) reste *inconnu* pour toujours.

Preuve : Supposons par absurde que p_i reçoit id_i (qu'il a envoyé au réveil). Ca implique que tout processus relaye id_i (par lignes 5, 6), mais c'est un absurde, car le processus le plus proche à p_i (suivant l'orientation) avec un id plus grande que id_i ne relayera pas id_i (il sera reçu, mais ignoré).

Théorème : Algorithme LCR (de base) résout le problème d'élection de leader (sans variantes).

Preuve : Lemme 2 et 3 prouvent les 2 condition de la spécification du problème (sans variantes).

Analyse de Complexité (LCR de base)
pire cas par défaut

- **En rounds synchrones ou asynchrone** (unités de temps)**,** jusqu'à la terminaison : O(n)
- **En messages** (pire cas) : $\Theta(n^2)$
  - Pire cas: les processus sont orientés dans l'ordre décroissant de leurs ids :
    $n + (n-1) + \ldots + 2 + 1 = n(n+1)/2$ messages
  - Meilleur cas : les processus sont orientés dans l'ordre croissant de leurs ids : 2n-1 messages
- **En bits** : $O(n^2 \log n)$, en supposant que tout id dans UID est de taille O(log n) bits

Élection de Leader (EL)
la spécification du problème
Variantes

Supposons que chaque nœud a une variable nommée *status* (état) t.q.
status ∈ {leader, inconnu, non_leader}

1. A la terminaison, exactement un processus a status = leader
2. Durant toute exécution, au plus un seul processus a status = leader

Les variantes :

A la terminaison (ou à partir d'une configuration), pour tout autre processus :

1. Il apprend de ne jamais être leader (status = non_leader)
2. il connaît l'id du leader
3. il détermine que le leader est déjà élu
4. le leader termine explicitement

Algorithme LCR - Variantes
pour un processus p_i

mon_id : UID ($\subseteq$ N+ - entiers positives) ;
état : {inconnu, leader, non-leader} ;
max_id : UID ;

mon_id := l'identificateur de pi
état := inconnu
max_id := mon_id

lors de réveil spontané →
1. envoyer (mon_id) à p_{i+1}

lors de réception de message j ≠ <termine> →
2. si (réveil provoqué) alors envoyer (mon_id) à p_{i+1}
3. si (mon_id = j) alors
4. état := leader
5. envoyer (<termine>) à p_{i+1}
6. sinon si (j > mon_id) alors
7. envoyer (j) à p_{i+1}
8. max_id := max(max_id, j)
fin si

lors de réception de message <termine> →
9. si (état ≠ leader) alors
10. état := non-leader
11. envoyer (<termine>) à p_{i+1}
fin si
12. terminer()

Algorithme de Hirschberg et Sinclair (HS)

Chaque processus p_i opère en **étapes** 0, 1, 2, ...

- Lors de chaque étape k, le processus p_i envoie deux *jetons (d'exploration)* contenant son identificateur id_i dans deux directions.
  - Ils sont supposés voyager à distance 2^k, puis retourner à leur origine p_i.
  - Si les deux jetons lui reviennent, le processus p_i passe a l'étape suivant k+1, i.e, ***reste actif***. Cependant il se peut que les jetons ne reviennent pas. Dans ce cas, le processus est dit ***éliminé***.
- Tant qu'un jeton est dans son phase d'exploration, chaque processus p_j sur son chemin compare id_i à son identificateur id_j.
  - Si $id_i < id_j$ le processus élimine le jeton (l'ignore) et
  - Si $id_i > id_j$ il le relaie.
  - Si $id_i = id_j$, le processus se déclare leader.
- Tous les processus relaient un jeton rentrant (à l'origine p_j).

HS : pseudo-code pour p_i

mon_id : UID ($\subseteq N^+$ - entiers positives) ; mon_id := l'identificateur de p_i
état : {inconnu, leader} ; état := inconnu

lors de réveil spontané →
1. envoyer ([*explore*, mon_id, **0**, 1]) à p_{i+1} et à p_{i-1}

lors de réception de [*explore*, j, k, d] de p_{i+1} (resp. p_{i-1}) →
2. si (réveil provoqué) alors envoyer ([*explore*, mon_id, **0**, 1]) à p_{i+1} et à p_{i-1}
3. si (mon_id = j) alors état := leader ; <terminer>
4. sinon si (j > mon_id et d < 2^k) alors
5. envoyer ([*explore*, j, k, **d+1**]) à p_{i-1} (resp. p_{i+1})
6. sinon si (j > mon_id et d ≥ 2^k) alors
7. envoyer ([*réponse*, j, k]) à p_{i+1} (resp. p_{i-1})

lors de réception de [*réponse*, j, k] de p_{i+1} (resp. p_{i-1}) →
8. si (j ≠ mon_id) alors
9. envoyer ([*réponse*, j, k]) à p_{i-1} (resp. p_{i+1})
10. sinon si (déjà reçu [*réponse*, j, k] de p_{i-1} (resp. p_{i+1})) alors
11. envoyer ([*explore*, j, **k+1**, **1**]) à p_{i+1} et à p_{i-1}

Complexité en messages de HS (1)

Dans chaque **étape k** :

1. **pour chaque processus actif** p_i, il y a au plus $\mathbf{4 \cdot 2^k}$ **jetons** avec l'id de p_i qui sont transmis sur l'anneau
   - Pour $k = 0$, n processus actifs
   - Pour $k \geq 1$, au plus $\lfloor n/(2^{k-1} + 1) \rfloor$ processus actif :
     - Car, la distance minimale entre 2 processus actifs en étape k est $2^{k-1} + 1$. Sinon, au moins un de ces deux processus n'aurait pas passé à l'étape k (aurait été éliminé à l'étape k-1)
2. alors, au plus $4 \cdot 2^k \cdot \lfloor n/(2^{k-1} + 1) \rfloor \leq 8n$ messages envoyés en étape $k \geq 1$; et $4n$ pour $k = 0$

Complexité en messages de HS (2)

- Le nombre d'étapes maximum jusqu'à l'élection est $\lceil \log n \rceil + 1$ (incluant l'étape 0)

➔ Au plus $8n\lceil \log n \rceil + 4n$ messages en tout

➔ **O(n log n)** messages

Complexité en temps de HS

- La dernière étape $\lceil \log n \rceil$ est de n rounds/unités de temps (synchrone/asynchrone)
- Une autre étape k dure au plus $2 \cdot 2^k$

➔Complexité en rounds/UTs :

$$2(2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^{\lceil \log n \rceil - 1}) + n =$$
$$= 2(2^{\lceil \log n \rceil} - 1) + n$$

- ➔ n est puissance de 2 : au plus 3n – 2 rounds
- ➔ n n'est pas puissance de 2 : au plus 5n – 2 rounds

➔**O(n)** rounds/unités de temps (synchrone/asynchrone)

Impossibilité d'EL sans identifiants

Théorème : Dans un anneau de taille n > 1, **si tous les processus sont identiques**, alors **il n'existe pas un algorithme *déterministe* d'élection de leader**, même si l'anneau est bidirectionnel (synchrone ou non) et même si la taille n est connu à chaque processus.

Preuve : Supposons par contradiction qu'un tel algorithme existe

- Considérons une exécution qui commence dans une configuration initiale C_0 où tout processus dans le même état s_0
- Supposons une exécution parfaitement synchrone t.q.
  pour chaque round r :
  - chaque processus envoie les mêmes messages aux mêmes voisins comme tout autre processus (tout p_i envoie message m_{r+} à p_{i+1} et m_{r-} à p_{i-1})
  - tout processus change son état de s_r à s_{r+1}
- Prouvons par récurrence sur chaque round r que chaque processus est dans le même état s_r que tout autre processus.
- Supposons que dans un round r^*, un leader est élu. Mais comme tout processus est dans le même état, alors tout processus est élu.

→ Ca contredit la spécification du problème d'EL.