Systèmes Distribués : Projet

Alain Cournier Stéphane Devismes

Résumé

Le but de ce projet est de programmer dans `Sinalgo` des algorithmes d'élection (au moins deux parmi ceux présentés dans les sections 2-5) pour des réseaux identifiés dont la topologie est un anneau dirigé (topologie `Circle` dans `Sinalgo`). Dans la suite, on dénote par $ID(p)$ l'identifiant du processus p.

- Le projet sera réalisé en binôme ou trinôme.
- Chaque groupe aura à programmer deux algorithmes obligatoires choisis par les responsables du cours. Les autres algorithmes pourront être programmés afin d'obtenir des points bonus.
- N'hésitez pas à poser vos questions aux responsables de cours, qui seront ravis d'y répondre.
- Vos fichiers source devront être commentés de manière judicieuse et détaillée (cela sera pris en compte dans la notation !)
- Vous devrez fournir une archive ZIP ou tgz contenant un rapport de 5 pages maximum et les sources de vos algorithmes.
- Le projet fera l'objet d'une soutenance de 30 min par groupe lors de la dernière séance.
- **DATE LIMITE DE RENDU : 25 janvier 2026 à 23h59**

1 Spécification de l'élection

De nombreux algorithmes répartis sont fondés sur l'hypothèse qu'il existe un processus distingué, appelé *leader* ou *racine*, qui aura un comportement différent des autres dans le protocole. C'est le cas, par exemple avec les algorithmes mono-initiateurs comme la circulation de jeton. Or, l'un des objectifs principaux dans la réalisation d'algorithmes distribués est de pouvoir interchanger le rôle d'un ou plusieurs composants du système. Donc, pour certains problèmes, il peut être nécessaire de distinguer dynamiquement au cours d'une exécution un (nouveau) processus parmi tous les processus du réseau : c'est le rôle d'un protocole d'élection.

Définition 1 (Spécification de l'élection)**.**

Sûreté (safety) *Au plus un processus est élu.*

Vivacité (liveness) *Un processus doit être élu en temps fini.*

2 Élection dans un anneau unidirectionnel (Algorithme de Le Lann)

Hypothèses :

- Processus et canaux asynchrones.
- Canaux FIFO.
- Processus avec identité.
- Au moins deux processus.
- Pas de faute.
- Topologie en anneau unidirectionnel : chaque processus distingue son voisin droit D et son voisin gauche G ; l'orientation est consistante. Voir figure 1.
- Multi-initiateur.

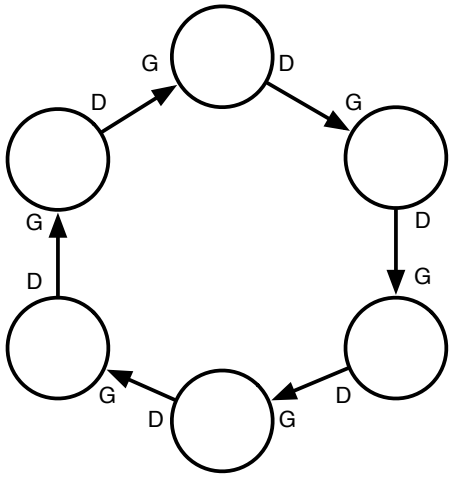

FIGURE 1 – Exemple d'anneau orienté.

L'algorithme. Chaque initiateur stocke dans une liste l'ensemble des identifiants des processus initiateurs, l'initiateur avec l'identifiant le plus petit est élu.

Chaque initiateur p émet un message $\langle Election, ID(p)\rangle$ et initialise sa liste avec son identifiant $ID(p)$. Le message $\langle Election, ID(p)\rangle$ est propagé sur l'anneau, toujours vers la droite : il est relayé par tous les autres processes. Lorsqu'un initiateur différent de p reçoit $\langle Election, ID(p)\rangle$, il ajoute $ID(p)$ à sa liste avant de relayer le message.

Les canaux sont supposés FIFO et un initiateur p génére son message $\langle Election, ID(p)\rangle$ avant qu'aucun autre message $\langle Election, ID(q)\rangle$ ne soit reçu. Donc, lorsqu'un initiateur p reçoit son message $\langle Election, ID(p)\rangle$, les messages $\langle Election, ID(q)\rangle$ de tous les autres processus initiateurs ont visité p. Ainsi, p détient tous les identifiants des initiateurs et peut alors choisir l'identifiant du processus élu (c'est-à-dire, le plus petit).

3 Élection dans un anneau unidirectionnel (Algorithme de Chang et Roberts)

Hypothèses :

- Processus et canaux asynchrones.
- Processus avec identité.
- Au moins deux processus.
- Pas de faute.
- Topologie en anneau unidirectionnel : chaque processus distingue son voisin droit D et son voisin gauche G ; l'orientation est consistante. Voir figure 1.
- Multi-initiateur.

L'algorithme. C'est une amélioration de l'algorithme de Le Lann. Chaque processus dispose d'un booléen *Leader* initialisé à $faux$.

- Chaque initiateur p envoie spontanément un message $\langle Election, ID(p)\rangle$ à son voisin de droite.
- Les suiveurs relaient à droite tous les messages qu'ils recoivent de la gauche.
- On retire de l'anneau les messages des processus dont on est sûr qu'ils vont perdre l'élection : un initiateur p relaie $\langle Election, ID(q)\rangle$ seulement si $ID(q) < ID(p)$.
- Un initiateur p affecte *Leader* à *vrai* s'il reçoit $\langle Election, ID(p)\rangle$.
- Chaque processus stocke l'identité d'initiateurs la plus petite connue.

4 Élection dans un anneau unidirectionnel (algorithme de Peterson)

Le but de cet algorithme est d'améliorer la complexité en nombre de messages de $O(n^2)$ à $O(n \log n)$.

Hypothèses :

- Processus et canaux asynchrones.
- Canaux FIFO.
- Processus avec identité.
- Au moins deux processus.
- Pas de faute.
- Topologie en anneau unidirectionnel : chaque processus distingue son voisin droit D et son voisin gauche G ; l'orientation est consistante. Voir figure 1.
- Multi-initiateurs.

L'algorithme. La particularité de cet algorithme est que le processus élu n'est pas nécessairement celui de plus petite (ou plus grande) identité.

Chaque processus p dispose d'un tag : une variable entière initialisée avec son identifiant $ID(p)$. Cependant, ce tag est amené à être modifier durant l'exécution.

Au cours de l'exécution, chaque processus est soit en mode actif soit en mode passif. Initialement, tous les processus initiateurs sont actifs, les autres sont passifs. Les processus actifs réalisent le vrai travail de l'algorithme, les passifs se contentent de relayer les messages. L'exécution se divise en phases.

Une phase se déroule de la manière suivante : chaque processus actif u_i envoie son tag aux deux processus actifs suivants dans l'anneau (des descendants donc). Lorsque u_i détient les tags des deux processus actifs le précédent dans l'anneau — disons u_{i-1} et u_{i-2} — il compare ces tags. Si u_{i-1} (son prédécesseur actif le plus proche) a le tag le plus petit des trois, alors u_i reste actif pour la phase suivante et adopte le tag de u_{i-1}. Dans, les autres cas, u_i devient passif. Enfin, lorsque lors d'une phase, un processus actif reçoit de son prédécesseur actif immédiat son propre tag, il en conclut qu'il est le dernier actif et se proclame élu.

Identifiants : Sous `Sinalgo`, les identifiants dans un anneau sont consécutif de 1 à n le long de l'anneau. Pour éviter les effets de bords dû à cette façon de numéroter les nœuds, vous utiliserez une autre variable, appelée `newID` comme identifiant. Cette variable sera initialisée dans la méthode `initialization()` comme suit :

```
newID = this.ID + ((int)(Math.random() * nbProcesses())) * nbProcesses().
```

5 Élection dans un anneau unidirectionnel (Algorithme d'Itai et Rodeh)

Hypothèses :

- Processus et canaux asynchrones.
- Canaux FIFO.
- **Processus anonymes.**
- Nombre de processus n connu.
- Au moins deux processus.
- Pas de faute.
- Topologie en anneau unidirectionnel : chaque processus distingue son voisin droit D et son voisin gauche G ; l'orientation est consistante. Voir figure 1.
- Multi-initiateur.

L'algorithme. Les processus ont deux états possibles : passif ou actif. Initialement, les initiateurs sont actifs et les suiveurs sont passifs. L'algorithme s'exécute par phase. Au début de la phase, les processus actifs tirent au hasard un tag (par exemple, entre 1 et n ou entre 1 et 2, ... [1]). À la fin de la phase, seuls les processus avec le plus petit tag reste actif. Chaque processus actif diffuse vers la droite un message contenant son tag, un booléen initialisé à $faux$ et un compteur initialisé à 1.

1. Les processus passifs relaient les messages reçus de la gauche vers la droite en incrémentant au préalable le compteur du message.
2. Si un actif reçoit un message contenant un tag plus grand que le sien, alors il ne relaie pas le message.
3. Si un actif reçoit un message contenant un tag plus petit que le sien, alors il devient passif. Il doit aussi relayer le message vers la droite toujours en incrémentant au préalable le compteur du message.
4. Lorsqu'un actif p reçoit un message avec son propre tag, on a plusieurs possibilité :
   (a) Si le compteur dans le message est inférieur à n (le nombre de processus), alors ce n'est pas le message initié par p. Dans ce cas, p relaie le message en ayant au préalable positionné le booléen à $vrai$ (pour signifier à l'initiateur du message que son tag n'est pas unique) et incrémenté le compteur.
   (b) Si le compteur dans le message est égal à n, alors la phase est terminée. Si le booléen du message est à $faux$, alors le processus est élu et l'élection est terminée (il est le dernier actif) ; sinon, le processus doit initier une nouvelle phase comme expliqué précédemment.

1. Testez plusieurs valeurs et remarquez l'impact sur la complexité.