Election dans un anneau : algorithme de Peterson

TD4 - Algorithmique distribuée - Polytech Grenoble INFO4

Le but de cet exercice est d'entraîner vos capacités d'analyse d'un algorithme donné : comprendre l'algorithme donné, démontrer quelques pas de la preuve, donner et démontrer sa complexité.

Hypothèses : Les processus et les communications sont asynchrones et les canaux de communication sont FIFO. Chaque processus est identifié avec un identifiant unique et chaque processus est initiateur. La topologie est un anneau unidirectionnel : chaque processus distingue son voisin droit et son voisin gauche, l'orientation étant consistante. Il n'y a pas de fautes.

Explications : Chaque processus est équipé de 3 variables. La variable *Output* représente le résultat calculé par l'algorithme (initialement à faux). La variable Tag est un entier représentant un identifiant de processus (initialement, Tag vaut l'identifiant du processus lui-même). Finalement, chaque processus est soit en mode actif soit en mode passif, ce mode étant représenté par la variable *Etat*. Les processus actifs réalisent le vrai travail de l'algorithme, les passifs se contentent de relayer les messages.

L'exécution se divise en phases successives. Lors d'une phase, chaque processus actif p envoie son tag aux deux processus actifs suivants dans l'anneau. Lorsque p détient les tags des deux processus actifs le précédant dans l'anneau, il compare les tags. Si son prédécesseur actif le plus proche, q, a le tag le plus petit des trois, p reste actif pour la phase suivante et adopte le tag de q. Dans, les autres cas, p devient passif. Lorsque un processus actif reçoit de son prédécesseur actif immédiat son propre tag, il positionne sa variable *Output* à vrai.

Question 1 – Que fait cet algorithme ? Spécifier le proprement en termes de propriétés de sûreté et de vivacité.

Question 2 – Cette question vous guide pour comprendre pourquoi l'algorithme est correct.

1. Deux processus consécutifs sont actifs au début d'une phase. Sont-ils toujours actifs à la fin de la phase ? Justifier.
2. Peut-il y avoir deux processus actifs portant le même tag ? Expliquer.
3. Se peut-il que tous les processus deviennent passifs ? Pourquoi ?

Conclure en justifiant les propriétés de sûreté et de vivacité.

Question 3 – L'algorithme a une complexité en $O(n \log n)$ messages, où n est le nombre de processus du réseau. Expliquer pourquoi.

Question 4 – Synthèse et conclusion : pourquoi cet algorithme est-il intéressant ?

Algorithme 1 Algorithme de Peterson pour un processus p

Variables

1: $Id_p \in \mathbb{Z}$ (identifiant unique du processus)
2: $Output \in \{Vrai, Faux\}$ (représente le résultat de l'élection : le processus est-il le leader ?)
3: $Etat \in \{actif, passif\}$
4: $Tag, V1, V2 \in \mathbb{Z}$

Algorithme pour tout processus

5: $Output \leftarrow Faux$
6: $Etat \leftarrow actif$
7: $Tag \leftarrow Id_p$
8: `Envoyer` $\langle Un, Tag \rangle$ `à` D
9: **Pour toujours**
10: `Réceptionner` $\langle Type, T \rangle$ `de` G
11: **Si** $Etat = passif$ **alors**
12: `Envoyer` $\langle Type, T \rangle$ `à` D
13: **SinonSi** $Type = Un$ **alors**
14: Traiter_Un(T)
15: **Sinon**
16: Traiter_Deux(T)
17: **Fin Si**
18: **Fin Pour toujours**

Fonction Traiter_Un(T)

19: **Si** $T = Tag$ **alors**
20: $Output \leftarrow vrai$
21: **Sinon**
22: $V1 \leftarrow T$
23: `Envoyer` $\langle Deux, T \rangle$ `à` D
24: **Fin Si**

Fonction Traiter_Deux(T)

25: $V2 \leftarrow T$
26: **Si** $V1 < V2 \wedge V1 < Tag$ **alors**
27: $Tag \leftarrow V1$
28: `Envoyer` $\langle Un, Tag \rangle$ `à` D
29: **Sinon**
30: $Etat \leftarrow passif$
31: **Fin Si**