Systèmes Distribués : TD 1, circulation d'un jeton dans un réseau quelconque

Alain Cournier Stéphane Devismes

1 Principe de l'algorithme

L'algorithme proposé par Tarry en 1895 résout le problème du labyrinthe. Représentons les intersections dans le labyrinthe par des sommets et les allées entre ces intersections par des arêtes. Un labyrinthe se modélise donc comme un graphe non orienté connexe. Enfin, utilisons un jeton pour symboliser le marcheur qui se déplace. L'algorithme de Tarry devient alors un algorithme de circulation de jeton dans un graphe quelconque. Le principe de l'algorithme de Tarry est le suivant : « Ne reprendre l'allée initiale qui a conduit à un carrefour pour la première fois que lorsqu'on ne peut pas faire autrement ». Ainsi, l'algorithme de Tarry applique le principe du parcours en profondeur d'abord.

2 Hypothèses

— Processus et canaux asynchrones.
— Canaux étiquetés de 1 à δ_p pour tout processus p.
— Pas de faute.
— Topologie quelconque (connexe) d'au moins deux nœuds.
— Mono-initiateur.

3 L'algorithme

Écrivez le code de l'algorithme. Pour cela, vous utiliserez un seul type de message : $\langle Jeton \rangle$; de plus vous utiliserez les variables locales (à chaque processus) suivantes.

— $Visite[1..\delta_p]$: tableau de Booléen, toutes les cases sont initialisées à $faux$.
— $Père \in \{\top, \bot\} \cup \{1 \dots \delta_p\}$: pointeur « parent », initialisé à $\bot$. Lors du démarrage « spontané », l'initiateur affecte sa variable $Père$ à $\top$.

4 Correction de l'algorithme

Question 1. Justifiez pourquoi il existe toujours au plus un jeton dans le réseau.

Par hypothèse, il y a un seul initiateur. Il exécute une seule fois le démarrage (« spontané ») de l'algorithme. Donc, un seul jeton est créé. Ensuite, les processus envoient un jeton seulement après l'avoir reçu. Par suite, il y a toujours au plus un jeton dans le réseau.

Question 2. Justifiez pourquoi au plus une décision est exécutée.

Il y a au plus une création de jeton car il y a un seul initiateur. Ensuite, il y a toujours au plus un jeton dans le réseau (*cf.* question précédente). Or, une décision est prise sur réception du jeton et dans ce cas, le jeton n'est pas renvoyé. Par suite, il y a au plus une décision.

Algorithme 1 *Circulation* pour tout processus p

Variables

1: $Visite[1..\delta_p]$: tableau de booléens indexé sur les numéros de canaux où chaque case est initialisée à $faux$
2: $Père \in \{1, ..., \delta_p\} \cup \{\bot, \top\}$ initialisé à $\bot$

Spontanément

3: $Père \leftarrow \top$
4: `Envoyer` $\langle Jeton \rangle$ `à` 1
5: $Visite[1] \leftarrow vrai$

Réception de $\langle Jeton \rangle$ **de** q

6: **Si** $Père = \bot$ **alors**
7: $\quad Père \leftarrow q$
8: **Fin Si**
9: **Si** $\forall q' \in \{1, ..., \delta_p\}, Visite[q'] = vrai$ **alors**
10: $\quad$ `décide`
11: **Sinon**
12: $\quad$ **Si** $\exists q' \in \{1, ..., \delta_p\} \mid Père \neq q' \wedge \neg Visite[q']$ **alors**
13: $\quad\quad$ **Si** $Père \neq q \wedge \neg Visite[q]$ **alors**
14: $\quad\quad\quad Suivant \leftarrow q$
15: $\quad\quad$ **Sinon**
16: $\quad\quad\quad$ Choisir $Suivant \in \{1, ..., \delta_p\}$ tel que $Père \neq Suivant \wedge \neg Visite[Suivant]$
17: $\quad\quad$ **Fin Si**
18: $\quad\quad$ `Envoyer` $\langle Jeton \rangle$ `à` $Suivant$
19: $\quad\quad Visite[Suivant] \leftarrow vrai$
20: $\quad$ **Sinon**
21: $\quad\quad$ `Envoyer` $\langle Jeton \rangle$ `à` $Père$
22: $\quad\quad Visite[Père] \leftarrow vrai$
23: $\quad$ **Fin Si**
24: **Fin Si**

Question 3. Combien de fois est traversée une arête lors de l'exécution ? (Justifiez).

Lemme 1. *Chaque canal est traversé au plus une fois dans chaque sens.*

Preuve. Après qu'un processus p ait envoyé le jeton dans un canal q, il affecte pour toujours $Visite[q]$ à $vrai$. Or, p n'envoie le jeton qu'à des canaux q tels que $Visite[q] = faux$. □

Question 4. Pourquoi l'exécution termine [1] toujours ?

D'après la question 3, chaque canal est traversé au plus une fois dans chaque sens. Ainsi, au plus $2m$ messages sont envoyés, où m est le nombre d'arêtes. Par suite, toute exécution termine.

Question 5. Pourquoi y a-t-il nécessairement une décision lors de l'exécution ?

Le jeton circule tant qu'il n'y a pas de décision. Donc d'après la question précédente au moins une décision finit par être prise.

Question 6. Quel processus finit par décider ? Pourquoi ?

Lemme 2. *L'initiateur finit par décider.*

Preuve. Le jeton circule tant qu'il n'y a pas de décision. Donc d'après le lemme précédent au moins une décision finit par être prise.

D'après la question 3, chaque processus envoie le jeton à travers chaque canal au plus une fois. Puisque le jeton est unique et créé par l'initiateur, chaque fois qu'un non-initiateur p détient le jeton, il l'a reçu une fois de plus qu'il ne l'a envoyé (faire un dessin). Ainsi, à la réception d'un jeton, le nombre de canaux par lesquels p a reçu le jeton est toujours supérieur au nombre de canaux par lesquels p a envoyé le jeton. Ainsi, à chaque réception du jeton, au moins un canal n'est pas marqué visité et par suite, p ne décide jamais. Donc seul l'initiateur peut décider. □

Question 7. Que peut-on dire des canaux du processus qui a décidé ?

Tous les canaux incidents au processus qui décide ont été traversés une fois dans chaque direction.

À la terminaison de l'algorithme, l'initiateur a envoyé le jeton à travers chacun de ses canaux, sinon il n'y aurait pas de décision.

De plus, d'après la question 3, l'initiateur a envoyé le jeton à travers chacun de ses canaux exactement une fois.

L'initiateur reçoit le jeton autant de fois qu'il l'a envoyé (le jeton est parti de lui !). Par suite, d'après la question 3, l'initiateur a reçu le jeton exactement une fois de chacun de ces canaux.

Question 8. Une fois la décision exécutée, que peut-on dire des canaux d'un processus visité ? En déduire qu'à l'exécution de la décision tous les processus ont été visité.

Pour tout processus visité p, tous les canaux incidents à p sont traversés une fois dans chaque direction.

Par contradiction. Soit p le premier (dans le temps) processus visité qui ne vérifie pas la propriété. D'après les réponses aux questions 6 et 7, p n'est pas l'initiateur. Ensuite, le père q de p vérifie la propriété, donc le lien (p, q) a été traversé une fois dans chaque sens. De plus, lorsque p envoie le jeton à q, p a envoyé le jeton une fois à tous ses autres voisins. p a reçu le jeton autant de fois qu'il l'a envoyé. Enfin, chaque voisin de p lui envoie le jeton au plus une fois d'après la question 3. Donc, chaque canal de p est traversé une fois dans les deux sens, contradiction.

Tous les processus finissent par être visité.

Si il y a des processus qui ne sont jamais visités, alors il existe deux voisins p et q tels que p est visité mais pas q (le réseau est connexe). Ce qui contredit le fait que chaque lien de p est traversé une fois dans les deux sens.

1. Une exécution termine si elle ne comporte qu'un nombre fini de réception de messages.

Question 9. Donnez la complexité en temps et en nombre de messages de l'algorithme.

$2m$, où m est le nombre d'arêtes.

5 Circulation perpétuelle

Question 10. Réécrivez l'algorithme pour en faire une circulation perpétuelle.

Algorithme 2 *Circulation* pour tout processus p

Variables

$Visite[1..\delta_p]$: tableau de booléens indexé sur les numéros de canaux où chaque case est initialisée à $faux$
$Père \in \{1, ..., \delta_p\} \cup \{\bot, \top\}$ initialisé à $\bot$

Spontanément

$Père \leftarrow \top$
`Envoyer` $\langle Jeton \rangle$ `à` 1
$Visite[1] \leftarrow vrai$

Réception de $\langle Jeton \rangle$ **de** q

Si $Père = \bot$ **alors**
    $Père \leftarrow q$
Fin Si
Si $\forall q' \in \{1, ..., \delta_p\}, Visite[q'] = vrai$ **alors**
    `décide`
    Pour tout $x \in \{1, ..., \delta_p\}$ **faire**
        $Visite[x] \leftarrow faux$
    Fin Pour
    `Envoyer` $\langle Jeton \rangle$ `à` 1
    $Visite[1] \leftarrow vrai$
Sinon
    Si $\exists q' \in \{1, ..., \delta_p\} \mid Père \neq q' \wedge \neg Visite[q']$ **alors**
        Si $Père \neq q \wedge \neg Visite[q]$ **alors**
            $Suivant \leftarrow q$
        Sinon
            Choisir $Suivant \in \{1, ..., \delta_p\}$ tel que $Père \neq Suivant \wedge \neg Visite[Suivant]$
        Fin Si
        `Envoyer` $\langle Jeton \rangle$ `à` $Suivant$
        $Visite[Suivant] \leftarrow vrai$
    Sinon
        `Envoyer` $\langle Jeton \rangle$ `à` $Père$
        Pour tout $x \in \{1, ..., \delta_p\}$ **faire**
            $Visite[x] \leftarrow faux$
        Fin Pour
        $Père \leftarrow \bot$
    Fin Si
Fin Si