

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*.
- $Père \in \{\top, \perp\} \cup \{1 \dots \delta_p\}$: pointeur « parent », initialisé à $\perp$. 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 où il émet un seul jeton. Ensuite, les processus envoient un jeton seulement après en avoir reçu un. 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 toujours au plus un jeton dans le réseau (*cf.* question précédente). De plus, un jeton ne peut être envoyé sans réception préalable que lors d'un démarrage (« spontané »). Or, il n'y a qu'un seul démarrage (« spontané ») puisqu'il n'y a qu'un seul initiateur. Par ailleurs, lorsqu'une décision est prise, elle intervient lors de la réception du jeton et en particulier après l'unique démarrage (« spontané »). De plus, dans ce cas, le jeton n'est pas renvoyé. Donc, après une décision, le système ne contient plus de jeton pour toujours. D'où, 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: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

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

Question 4. Pourquoi l'exécution termine¹ 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. 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.

Or, d'après la question 3, chaque voisin de p envoie le jeton à p au plus une fois.

Ainsi, à la réception d'un jeton, $\delta_p \geq \#R > \#E$ où $\#R$ est le nombre de canaux par lesquels p a reçu le jeton et $\#E$ est le nombre de canaux par lesquels p a envoyé le jeton.

D'où, à chaque réception du jeton, on a $\#E < \delta_p$, c'est-à-dire qu'au moins un canal n'est pas marqué visité. Par suite, p ne décide jamais. Donc, seul l'initiateur peut décider. Or, il finit par le faire, d'après la question précédente. $\square$

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

Tous les canaux incidents au processus qui a décidé 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 visités.

1. Une exécution termine si elle atteint en temps fini une configuration à partir de laquelle le système contient plus jamais de message. Sous l'hypothèse de canaux fiables, une exécution termine si elle ne comporte qu'un nombre fini d'envoi/réception de messages

Une fois la décision exécutée, pour tout processus visité p , tous les canaux incidents à p ont été 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. Or, lorsque p envoie le jeton à q , p a envoyé le jeton une fois à tous ses autres voisins, d'après l'algorithme et la question 3. Donc, à la décision, p a envoyé le jeton une fois à tous ses voisins, toujours d'après la question 3. De plus, par unicité du jeton (Question 1), à la décision à l'initiateur (Question 1), le non-initiateur p a reçu le jeton autant de fois qu'il l'a envoyé, donc δ_p fois. De plus, chaque voisin de p lui a envoyé le jeton au plus une fois d'après la question 3. Donc, lors de la décision, chaque canal de p a été traversé une fois dans les deux sens, contradiction.

À l'exécution de la décision, tous les processus ont été visités.

Si il y a des processus qui n'ont jamais été visités, alors il existe deux voisins p et q tels que p a été visité mais pas q (le réseau est connexe). Ce qui contredit le fait que chaque canal de p a été traversé dans les deux sens.

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

$2m$, où m est le nombre d'arêtes. Cf. Questions 1 et 8.

5 Circulation perpétuelle

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

Algorithme 2 *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: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

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** $Père = \perp$ **alors**
 - 7: $Père \leftarrow q$
 - 8: **Fin Si**
 - 9: **Si** $\forall q' \in \{1, \dots, \delta_p\}, Visite[q'] = vrai$ **alors**
 - 10: décide
 - 11: **Pour tout** $x \in \{1, \dots, \delta_p\}$ **faire**
 - 12: $Visite[x] \leftarrow faux$
 - 13: **Fin Pour**
 - 14: **Envoyer** $\langle Jeton \rangle$ à 1
 - 15: $Visite[1] \leftarrow vrai$
 - 16: **Sinon**
 - 17: **Si** $\exists q' \in \{1, \dots, \delta_p\} \mid Père \neq q' \wedge \neg Visite[q']$ **alors**
 - 18: **Si** $Père \neq q \wedge \neg Visite[q]$ **alors**
 - 19: $Suivant \leftarrow q$
 - 20: **Sinon**
 - 21: Choisir $Suivant \in \{1, \dots, \delta_p\}$ tel que $Père \neq Suivant \wedge \neg Visite[Suivant]$
 - 22: **Fin Si**
 - 23: **Envoyer** $\langle Jeton \rangle$ à $Suivant$
 - 24: $Visite[Suivant] \leftarrow vrai$
 - 25: **Sinon**
 - 26: **Envoyer** $\langle Jeton \rangle$ à $Père$
 - 27: **Pour tout** $x \in \{1, \dots, \delta_p\}$ **faire**
 - 28: $Visite[x] \leftarrow faux$
 - 29: **Fin Pour**
 - 30: $Père \leftarrow \perp$
 - 31: **Fin Si**
 - 32: **Fin Si**
-