Systèmes Distribués : TD 2, Propagation d'Information avec Retour (PIR) dans un réseau quelconque

Alain Cournier Stéphane Devismes

Résumé

La propagation d'information avec retour (PIR) est un mécanisme de diffusion avec accusés de réception. C'est un outil de base en algorithmique distribuée, notamment pour calculer des infimums, faire de la prise d'instantanée, ou encore détecter la terminaison d'un autre algorithme.

1 Les hypothèses

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

2 Le principe de l'algorithme

Dans cet algorithme, chaque processus va maintenir deux variables :

— Un pointeur de canal *Père*, initialisé à $\perp$. Sur le modèle de la circulation de jeton, l'initiateur devra fixer son pointeur à $\top$. Pour chaque suiveur, *Père* désignera l'arête incidente le reliant à son père dans l'arbre couvrant construit durant la première phase de l'algorithme (la phase de diffusion).

— Une variable *Cpt*, initialisée à 0. Ce compteur contiendra le nombre de messages reçus par le processus.

L'algorithme s'exécute en deux phases : la phase de diffusion et la phase de retour.

L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser. Dans ce cas, il initie la diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

Lorsqu'un processus p reçoit un message *Brd* de q, il incrémente son compteur. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception, il affecte son pointeur *Père* à q et relaie le message à tous ses voisins sauf q. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR : si *Cpt* est supérieur ou égal à son degré δ_p. Dans ce cas, si p est l'initiateur, il décide, sinon il envoie un accusé réception *Ack* à son père (phase de retour).

Lorsqu'un processus p reçoit un message *Ack* de q, il incrémente son compteur. Puis, il teste s'il a terminé sa participation au PIR. Dans ce cas, si p est l'initiateur, il décide, sinon il envoie un accusé réception *Ack* à son père.

3 La spécification du PIR

D'après vous, quelle est la spécification de cet algorithme à vague ?

Définition 1 (Spécification du PIR de la donnée d)**.**

Sûreté

1. *Au plus une décision est prise.*
2. *Si une décision est prise, alors elle dépend causalement d'un accusé de réception de d envoyé par chaque processus suiveur.*

Vivacité

1. *L'exécution termine.*
2. *Tout processus suiveur reçoit d.*
3. *Tout processus suiveur accuse réception de d.*
4. *L'initiateur finit par décider.*

4 L'algorithme

Écrivez le code de l'algorithme.

Algorithme 1 *PIR* de la donnée d pour tout processus p

Variables

1: $Père \in \{1, ..., \delta_p\} \cup \{\bot, \top\}$ initialisé à $\bot$
2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$
4: **Pour tout** $q \in \{1, ..., \delta_p\}$ **faire**
5: `Envoyer` $\langle Brd,d \rangle$ `à` q
6: **Fin Pour**

Réception de $\langle Brd,d \rangle$ **de** q

7: $Cpt \leftarrow Cpt + 1$
8: **Si** $Père = \bot$ **alors**
9: $Père \leftarrow q$
10: **Pour tout** $v \in \{1, ..., \delta_p\} \setminus \{q\}$ **faire**
11: `Envoyer` $\langle Brd,d \rangle$ `à` v
12: **Fin Pour**
13: **Fin Si**
14: `Retour()`

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

15: $Cpt \leftarrow Cpt + 1$
16: `Retour()`

Fonction `Retour()`

17: **Si** $Cpt \geq \delta_p$ **alors**
18: **Si** $Père = \top$ **alors**
19: `décide`
20: **Sinon**
21: `Envoyer` $\langle Ack \rangle$ `à` $Père$
22: **Fin Si**
23: **Fin Si**

5 La correction de l'algorithme

Question 1. Justifiez pourquoi tous les processus suiveurs reçoivent la donnée d.

Lemme 1. *Tout processus suiveur reçoit un message Brd.*

Preuve. Supposons, par contradiction, qu'un processus suiveur ne reçoive aucun message Brd.

A l'initialisation, l'initiateur envoie Brd à tous ses voisins (qui sont suiveurs). Donc, tous les voisins de l'initiateur reçoivent Brd. Comme le réseau est connexe, il existe au moins un processus suiveur p qui reçoit Brd et qui est voisin d'un processus suiveur q qui ne le reçoit pas.

Soit r le processus émetteur du message Brd que p reçoit en premier. Soit r est l'initiateur, soit r a reçu Brd au préalable. Donc $r \neq q$. Or, lorsque p reçoit pour la première fois Brd, p envoie Brd à tous ses voisins sauf r. Donc q finit par recevoir Brd, contradiction. □

Corollaire 1. *Tout processus suiveur reçoit d.*

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal. De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal. Dans la suite, on dira que p *désigne* q *comme père* si la variable *Père* de p pointe le canal reliant p à q.

Soit $T = (V, E)$ un graphe orienté où V est l'ensemble des processus et E est l'ensemble des arêtes telles que $(p, q) \in E$ si et seulement si $p, q \in V$ et p finit par désigner q comme père. Tout d'abord, $|E| = n - 1$. Ensuite, pour tout processus p, il existe un chemin (orienté) dans T de p vers l'initiateur. D'où :

Lemme 2. *T est un arbre couvrant orienté enraciné à l'initiateur.*

Dans la suite, nous pourrons utiliser T dans le raisonnement.

Question 2. Prouvez que tout processus envoie au moins un message à chacun de ses voisins.

Lemme 3. *Tout processus envoie au moins un message à chacun de ses voisins.*

Preuve. Supposons que le processus p n'envoie aucun message à l'un de ses voisins.

Tout d'abord, p n'est pas l'initiateur. Ensuite, d'après la question 1, p finit par recevoir Brd. Lorsqu'il reçoit Brd pour la première fois, il le relaie à tous ses voisins sauf son père. Donc p n'envoie aucun message à son père.

Soit $p_{\max}$ un processus qui n'envoie aucun message à son père mais tel que tous les membres de son sous-arbre $T(p_{\max})$ (le sous-arbre de T enraciné en $p_{\max}$) le font.

Tous les voisins de $p_{\max}$ qui ne sont pas ses fils dans T (dont son père) finissent par recevoir Brd (lemme 1) et dans ce cas relaient Brd en particulier à p_{max}. Donc p_{max} reçoit au moins un message de chacun de ses voisins et ainsi finit par envoyer un message Ack à son père, contradiction. □

Question 3. Justifiez pourquoi l'initiateur finit par décider.

D'après la question précédente, l'initiateur i reçoit au moins δ_i messages, d'où :

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

Question 4. Justifiez pourquoi tout processus suiveur accuse réception de d (Ack).

Un processus suiveur n'envoie pas Brd à son père, or d'après la question 2, il envoie au moins un message à son père. D'où, il envoie nécessairement Ack à son père et on a le corollaire suivant :

Corollaire 3. *Tout processus suiveur accuse réception de d (Ack).*

Question 5. Justifiez pourquoi l'exécution termine.

Lemme 4. *Tout processus envoie au plus un message à chacun de ses voisins.*

Preuve.

1. Tout processus envoie 0 message Brd à son père dans T et 1 message Brd à ses autres voisins.
2. Tout processus p envoie Ack uniquement à son père et seulement si p n'est pas initiateur.
3. Supposons, par contradiction, qu'un processus envoie Ack au moins deux fois à son père. En particulier, c'est suiveur.

   Soit p un processus suiveur qui envoie deux fois Ack à son père, mais tel que tous les membres de sont sous-arbre $T(p)$ envoient Ack au plus une fois à leur père.

   Les fils de p ne lui envoie jamais Brd (Point 1), donc p reçoit au plus un message de chacun de ses fils. Ensuite, les voisins non fils de p ne lui envoie jamais Ack (Point 2) et au plus un message Brd (Point 1). Donc, p reçoit au plus un message de chacun de ses voisins. Ainsi, p envoie au plus une fois Ack à son père, contradiction.

D'après les trois points précédents, le lemme est vérifié. □

Corollaire 4. *L'exécution termine.*

Question 6. Justifiez pourquoi au plus une décision est prise.

Seul l'initiateur peut décider. De plus, d'après la réponse à la question précédente, il décide au plus une fois car il reçoit au plus un message de chacun de ses voisins.

Question 7. Justifiez pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus.

Lemme 5. *Si une décision est prise, alors elle dépend causalement d'un accusé de réception de chaque processus.*

Preuve. L'initiateur a au moins un fils dans T. D'après les questions précédentes, l'initiateur décide après avoir reçu Ack de chacun de ses fils dans T. Donc la décision de l'initiateur dépend causalement des accusés de réception de ses fils dans T.

Par récurrence sur $h(p)$ (la hauteur de p), nous pouvons prouver que le message Ack envoyé par tout suiveur envoie à son père dépendent causalement d'un accusé de réception de chacun des processus de son sous-arbre. Par suite, le lemme est vérifié. □

D'après l'ensemble des réponses aux questions, vous pouvez conclure :

Théorème 1. *L'algorithme 1 résout la propagation d'information avec retour dans un réseau quelconque.*

Remarque 1. *Pour pouvoir répéter les vagues, il suffit de réinitialiser les variables après l'envoi de l'accusé de réception et après la décision.*

Remarque 2. *Pour faire du multi-initiateur, il faut dupliquer l'algorithme en n algorithmes (n étant le nombre de processus) et taguer les messages et variables avec les identités.*

6 La complexité de l'algorithme

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les lemmes précédents, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

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

La phase de diffusion s'exécute en au plus $\mathcal{D}$ unités de temps, où $\mathcal{D}$ est le diamètre du réseau. Ensuite, la phase de retour consiste en une propagation de bas en haut de l'arbre. La hauteur de l'arbre étant au maximum $n-1$, cela se fait en au plus $n-1$ unités de temps. D'où, une complexité en temps de $\mathcal{D}+n-1 \leq 2n-2 \leq 2m$.