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 ?

4 L'algorithme

Écrivez le code de l'algorithme.

Algorithme 1 PIR de la donnée d pour tout processus p

5 La correction de l'algorithme

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

Preuve. Par définition, T contient un arc par suiveur donc $|E| = n - 1$.

Considérons maintenant la relation R entre couples de processus définie comme suit : *q R p signifie que (1) p et q sont des processus suiveurs et (2) p a reçu un message pour la première fois par q*; en particulier, cela signifie que q est le père de p. R est irréflexive car un processus ne s'envoie pas de message à lui-même. Soit t_r l'instant où un processus suiveur r reçoit un message pour la première fois. t_r est bien défini car r reçoit au moins un message, d'après la réponse de la question 1. On a *(*) pour tous processus p et q tels que q R p*, $t_q < t_p$ car q étant un suiveur il envoie un message à p seulement après avoir reçu lui-même un message.

Considérons ensuite la relation $\prec$ obtenue par fermeture transitive de R. Par *(*)* et transitivité, on a *(**) pour tous processus p et q tels que $q \prec p$, on a $t_q < t_p$*.

Montrons, par contradiction, que $\prec$ est un ordre partiel strict (c'est-à-dire, une relation irréflexive, transitive et asymétrique [1]) : Si $\prec$ n'est un ordre partiel strict alors il existe une suite de processus suiveurs $p_0, \ldots p_{k-1}$ tel que $\forall i \in [0..k-1]$, $p_i \prec p_{(i+1) \bmod k}$. Donc, on a $\forall i \in [0..k-1]$, $t_{p_i} < t_{p_{(i+1) \bmod k}}$ par *(**)*. Par transitivité, on a $t_{p_0} < t_{p_k}$. Or, $p_k \prec p_0$ implique $t_{p_k} < t_{p_0}$, contradiction. Donc, $\prec$ est bien un ordre partiel strict.

Considérons maintenant un ordre total sur les processus $p_1 \ldots, p_n$ compatible avec $\prec$ où p_1 est l'initiateur (il est bien défini car on n'a jamais p_i R p_1). Prouvons par récurrence sur i que pour tout processus p_i avec $i \in [1..n]$, il existe un chemin (orienté) dans T de p_i vers l'initiateur.

Tout d'abord, le cas de base $i = 1$ est trivial : le chemin vide relie p_1 à lui même (dans T). Soit $i > 1$. Puisque p_i est un suiveur, posons p_j le père de p_i. Par définition de l'ordre, on a $j < i$ car soit p_j est l'initiateur soit p_j R p_i et donc $p_j \prec p_i$. Donc, par hypothèse de récurrence, il existe un chemin (orienté) P dans T de p_j vers l'initiateur. Or, par définition, l'arc (p_i, p_j) appartient à E. Donc, p_i, P est un chemin (orienté) dans T de p_i vers l'initiateur et la récurrence est vérifiée.

Ainsi, nous pouvons conclure que T est 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.

1. Notez que irréflexive et transitif implique asymétrique.

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

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

Question 5. Justifiez pourquoi l'exécution termine.

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

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

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.

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