Séquence 2 - Détection de terminaison
Jean.Saquet@info.unicaen.fr

,

1 Définition et propriétés du réseau

On considère un réseau de sites communicants par messages et ayant les propriétés suivantes :
– Les lignes sont fiables : les messages ne sont ni perdus, ni altérés, ni dé-séquencés.
– Les temps de transmissions des messages sont finis.
– Les sites sont numérotés 1..n
– La topologie du réseau est quelconque. Chaque site connaît l'ensemble de ses voisins. (constante Voisinsi)
– Un des sites joue un rôle particulier (l'initiateur, numéroté 0, cf. ci-dessous)
On va construire un algorithme de détection de terminaison, en partant de la spécification abstraite de la propriété de terminaison.
On suppose donc qu'un algorithme réparti fonctionne sur le réseau, il sera référencé ci-dessous sous l' appellation"algorithme de base". Vis à vis de cet algorithme, les différents sites du réseau peuvent être actifs ou passifs. Cet état est mémorisé dans la variable $Etat_i$. Ils ne peuvent envoyer des messages relatifs à l'algorithme de base que s'ils sont actifs. Par contre, la réception d'un message relatif à cet algorithme les rend actifs (s'ils ne l'étaient pas déjà). Pour simplifier le problème, le site 0 (l'initiateur) est supposé être toujours passif (il ne participe pas à l'algorithme de base).

2 Définition de la terminaison

La propriété de terminaison peut être définie comme suit :

Terminé = conjonction des P_i

Où :
P_i= ($Etat_i$ = passif) et ($\forall j \in Voisins_i$, le canal de i vers j est vide).

(on distingue ici le canal de i vers j et le canal de j vers i : l'un peut être vide et l'autre pas).
L'idée de base pour la construction d'un algorithme de terminaison est que chaque site i évalue localement le prédicat P_i (i $\neq$ 0), et que le site 0 collecte les résultats de ces évaluations.

3 Problèmes liés au caractère réparti du système

– Le site i contrôle l'envoi de ses messages, mais pas leur réception. Il ne peut donc pas savoir si les canaux vers ses voisins sont vides ou non.
– La collecte des résultats ne peut pas se faire instantanément ni simultanément sur tous les sites. Certains d'entre eux peuvent changer d'état pendant cette collecte.

On va traiter les deux problèmes successivement.

4 Problème de la détection de la réception des messages

⇒ **1.** *Proposer une solution afin qu'un site sache que les messages qu'il a envoyés ont été reçus et traités par les destinataire. Remplacer dans la définition de la terminaison les prédicats P_i par des prédicats Q_i tels que la conjonction de ces derniers entraîne celle des P_i, et que si tous les P_i sont vrais, alors les Q_i le deviennent en un temps fini.*

5 Principe de l'algorithme réparti

On utilise en fait un parcours afin de demander à chaque site de nous envoyer son état. Le site initiateur pourra également relancer un parcours si le résultat du précédent ne lui a pas permis de détecter la terminaison de l'algorithme de base. On appellera ceci des "vagues" de demande d'état. On ne détaillera pas dans le cas général ce parcours ni la manière dont chaque site envoie la réponse à l'initiateur, on supposera seulement que ces vagues sont telles que, lorsque le site 0 lance une vague de numéro n, cette vague visitera tous les sites du réseau avant de revenir au site initiateur, avec un paramètre booléen v qui sera la conjonction des prédicats collectés dans les différents sites. Les algorithmes de parcours vus à la séance 1 permettent ceci (version collecte d'informations).

6 Problème de la synchronisation des informations envoyées

Afin de résoudre la deuxième difficulté, on va remplacer le prédicat Q_i par le prédicat R_i suivant :

R_i = le site i est resté passif entre la visite courante de la vague et la précédente

On suppose donc que le paramètre v de retour_vague contient la conjonction des R_i. n est le numéro de vague.
On va donc gérer sur chaque site i une variable $Cont\_passif_i$, réalisant le prédicat R_i, de la manière suivante :
- $Cont\_passif_i$ est initialisée à faux.
- Lorsque i reçoit un message de l'algorithme de base, $Cont\_passif_i$ est mis à faux (puisque le site redevient actif)
- Lorsque i envoie un message de l'algorithme de base, aucune action sur $Cont\_passif_i$, puisque le site est nécessairement actif.
- Lorsque i reçoit Visite_Vague(n,v), i attend que Q_i soit vrai, puis :
  - exécute v := v et $Cont\_passif_i$
  - met $Cont\_passif_i$ à vrai
  - appelle Faire_Suivre_vague(n,v)

⇒ **2.** *Écrire les algorithmes complets dans le cas où il existe une structure d'anneau permettant de simplifier le mécanisme de vague, ainsi que dans le cas où il existe une structure d'arbre couvrant.*

7 Propriétés de l'algorithme obtenu

⇒ **3.** *Prouver que lorsque l'algorithme détecte une terminaison, l'algo de base est bien terminé, et que, inversement, si l'algo de base se termine, notre algorithme détectera cette terminaison au bout d'un temps fini.*

Indication : numéroter les vagues et considérer l'instant où la vague numéro est propagée par le site i, celui où elle se termine et les états de chaque variable $Cont\_passif_i$ à ces instants, en fonction du fait que l'algorithme de base est terminé ou non.