Université de CAEN
U.F.R. de Sciences

Licence d'informatique
Semestre 6- E I61 - Algo

Séquence 2 - Détection de propriétés globales

Jean.Saquet@info.unicaen.fr

1 Définition et propriétés du réseau

Définitions et hypothèses :
Un état global d'un système réparti est constitué :
– d'un état pour chaque site i du système, constitué par l'ensemble des valeurs des variables locales à ce site, et qu'on représentera par El_i.
– d'un contenu pour chaque canal de communication, constitué d'une file de messages, éventuellement vide.
Ainsi défini, un état global quelconque ne correspond pas nécessairement à un état réellement atteint par le système à un moment donné, ni même à un état possible à atteindre au cours d'une exécution de l'algorithme réparti fonctionnant sur ce système. Ci-dessous, l'algorithme réparti dont on cherchera à observer un état global sera appelé algorithme de base.

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.
– Un des sites joue un rôle particulier (l'initiateur, cf. ci-dessous) et tous les sites disposent d'une primitive permettant d'envoyer un message à l'initiateur en un temps fini. L'initiateur connaît tous les sites du réseau.
– Tout site possède des canaux d'entrée et des canaux de sortie vers d'autres sites. Ces canaux sont unidirectionnels.
– Le réseau est fortement connexe (pour tout couple i,j de sites, il existe au moins un chemin de i vers j).
– L'algorithme réparti de base fonctionnant sur ce système est quelconque.

2 Présentation du problème et principe de la solution

On se propose d'écrire un algorithme réparti permettant à un site, l'initiateur, d'obtenir un état global du système. Le principe est le suivant :

L'initiateur lance un parcours de réseau (parallèle) demandant à chaque site de mémoriser son état, en vue d'un envoi ultérieur. Lors de la réception de ce message, chaque site enregistre son état dans une variable de sauvegarde, et transmet le message selon le principe de la diffusion contrôlée vue au cours 1. Selon ce principe, chaque site va en fait recevoir le message par chacun de ses canaux d'entrée. Lorsqu'il les aura tous reçus, il enverra à l'initiateur, via son père dans l'arbre couvrant construit, l'état qu'il avait sauvegardé, mais aussi, pour chaque canal d'entrée, un certain nombre de messages de l'algorithme de base.

Quels sont ces messages qu'il faut mémoriser et envoyer à l'initiateur ? Ce sont ceux qui étaient dans les canaux de communication au "moment" de la "photographie" de l'état du réseau, c'est-à-dire tous ceux qui ont été envoyés par un site i AVANT la mémorisation de son état local, donc avant que i ait reçu le premier message de demande de l'initiateur, et reçus par un site j APRES la mémorisation de son état local, donc après la réception par j du premier message de l'initiateur.

Il est bien évident que les canaux de communication ne peuvent pas fournir eux-mêmes leur contenu. Il faudra donc attendre que chaque site ait reçu les messages concernés pour les envoyer à l'initiateur. C'est

pour cette raison que les sites ne répondent pas immédiatement à sa requête, mais seulement lorsqu'ils ont pu enregistrer toutes les données nécessaires.

Pour bien repérer les messages concernés, on peut utiliser les horloges logiques de Lamport. En effet, si tous les messages, de l'algorithme de base et de celui d'obtention d'état, sont datés de cette manière, il suffira de considérer, sur chaque site i et sur chacun de ses canaux d'entrée x, les messages de l'algorithme de base qui ont une date :
– postérieure à la date de réception par i du premier message de demande d'état de l'initiateur
– antérieure à la date de ce même message, retransmis par le voisin de i sur le canal d'entrée x de i.

3 Écriture de l'algorithme

En fait, on peut se passer des horloges logiques en simplifiant un peu. En effet, il faut enregistrer les messages de l'algorithme de base reçus par i après qu'il ait reçu le premier message de demande de l'initiateur et, pour chacun de ses canaux d'entrée, avant le même message renvoyé sur ce canal. Ces messages de demande jouent donc le rôle de "marqueur" qui délimitent le temps pendant lequel on enregistre les messages à renvoyer à l'initiateur.

L'algorithme d'obtention d'un état global du système sera donc basé sur les principes suivants :
– Les messages utilisés par cet algorithme sont du type DemandeEtat (sans paramètres) ou IndiqueEtat (avec pour paramètres l'identité i du site emetteur, une copie de son état local Eli enregistré à la réception du premier message DemandeEtat, et des files indiquant le contenu de chacun de ses canaux d'entrée).
– Un seul site du réseau, l'initiateur, peut lancer l'algorithme d'obtention de l'état global, en envoyant des messages de type DemandeEtat sur tous ses canaux de sortie. Ces messages doivent être propagés par les autres sites de manière à ce que tout canal de communication transmette une et une seule fois un message de type DemandeEtat au cours du déroulement de l'algorithme.
– Tout site (y compris l'initiateur) enregistre son état dans une variable SvgEtat à la réception du premier message du type DemandeEtat. A partir de cet instant, le site enregistre dans une file EtatCanal[k] une copie de tout message de travail arrivant sur le canal d'entrée k, et ceci jusqu'au moment ou il reçoît un message DemandeEtat sur le canal k.
– Lorsqu'un site a reçu un message de type DemandeEtat sur chacun de ses canaux d'entrée, il transmet au site initiateur l'état local sauvegardé SvgEtat, ainsi que les files EtatCanal[x], pour tout canal d'entrée x, ceci dans un message de type IndiqueEtat.
– Lorsque l'initiateur à reçu un message IndiqueEtat de tout site du réseau (y compris de lui-même), l'obtention de l'état global est terminé et il doit pouvoir lancer à nouveau le même algorithme.

L'algorithme détaillé peut s'écrire ainsi :

```
Variables de chaque site i :
        Enregistre booléen initialisé à faux
        Envoye booléen initialisé à faux
        SvgEtat (destiné à recevoir la copie de l'état local)
        pour tout canal d'entrée y :
                Recu[y] booléen initialisé à faux
                Etatcanal[y] file de messages

Initiateur :
-sur décision de lancement du calcul :
        Enregistre := vrai; Envoye := faux
        copier l'état local Eli dans SvgEtat
        pour tout canal d'entrée y, faire EtatCanal[y] := vide
        pour tout canal de sortie x, envoyer DemandeEtat sur x
-sur réception de IndiqueEtat(j, El, suite de files de messages)
        mémorise l'état local du site j pour utilisation ultérieure
```

```
Tout site :
-sur réception de DemandeEtat venant du canal d'entrée y
        Si Enregistre = faux
        alors
                Enregistre := vrai; Envoye := faux; Pere := y
                copier l'état local Eli dans SvgEtat
                pour tout canal d'entrée z faire
                        EtatCanal[z] := vide
                        Recu[z] := faux
                Recu[y] := vrai
                pour tout canal de sortie x, envoyer DemandeEtat sur x
        sinon Recu[y] := vrai

         Si (Enregistre = vrai) et (Envoye = faux) et
                   (pour tout canal d'entrée x, recu[x] = vrai)
         alors
                     Enregistre := faux
                     Envoye := vrai
                     Envoyer IndiqueEtat(i, SvgEtat, {EtatCanal[y] / y canal d'entrée}
                         à l'initiateur via Pere (éventuellement soi-même)

-sur réception d'un message m de l'algorithme de base, sur le canal d'entrée y
Si (Enregistre = vrai) et (Recu[y] = faux)
        alors ajouter le message m à la file EtatCanal[y]
```

Commentaires :
- Tout site envoie une fois et une seule le message DemandeEtat sur chacun de ses canaux de sortie (la variable Enregistre sert à mémoriser le fait que ce message à été reçu et propagé)
- L'état local de chaque site est enregistré à la première réception de DemandeEtat
- Les messages de l'algorithme de base, reçus sur un canal d'entrée y, sont enregistrés dans la file EtatCanal correspondant au canal d'entrée y s'ils arrivent entre le moment où le site a reçu le premier DemandeEtat (Enregistre est à vrai), et celui où il reçoît le message DemandeEtat sur le canal y (Recu[y] est à faux).
- Lorsqu'un site à reçu DemandeEtat sur chacun de ses canaux d'entrée, tous ses Recu[y] sont à vrai et il envoie son état SvgEtat ainsi que ses files EtatCanal[y] à l'initiateur.
- La variable Envoye sert a distinguer les différentes phases de l'algorithme (demandes successives faites par l'initiateur). Elle n'est pas indispensable mais permet de clarifier les choses. L'initiateur ne doit lancer une nouvelle demande que s'il a reçu une réponse de tous les sites du réseaux (ce qui correspond au fait que toutes les variables Envoye sont à vrai).

4 Application

En utilisant l'algorithme vu ci-dessus, on peut par exemple écrire un algorithme de détection de terminaison fonctionnant sur le site initiateur en supposant que pour tout site, un champ de l'état local indique si ce site est actif ou passif relativement à l'algorithme de base dont on veut détecter la terminaison, et que le site initiateur est capable de déterminer si un message est relatif à cet algorithme ou pas. En voici le texte :

```
Algorithme de l'initiateur :
Terminaison := faux
Répéter
        Lancer le calcul d'état global {en utilisant l'alg. de la question 1}
        Attendre un IndiqueEtat de chaque site du réseau
        Si tous les sites sont passifs, et si les files représentant les états des canaux de
                 comm. ne contiennent aucun message relatif à l'algorithme de base
        alors Terminaison := vrai
jusqu'à Terminaison = vrai
```

On remarque que cet algorithme nécessite que l'initiateur connaisse tous les sites du réseau pour détecter le fait qu'ils ont tous répondu à sa demande.