TD sur les horloges logiques.

Exclusion mutuelle de Ricart et Agrawala

Nous avons vu en cours un algorithme d'exclusion mutuelle qui modélise une file d'attente répartie en utilisant les horloges de Lamport. Le principe de l'algorithme est le suivant : chaque site désirant la ressource va demander la permission à tous les autres. On départage les conflits en étiquetant chaque demande par l'heure logique à laquelle on a fait cette demande.

Exercice : Section critique à entrées multiples

On considère le problème de la k-exclusion mutuelle : une ressource qui doit être accédée en exclusion mutuelle est disponible en k exemplaires. Toutes les demandes des processus ne portent que sur un exemplaire de la ressource et un processus n'émet pas de nouvelle demande tant que sa demande en cours n'a pas été satisfaite. On peut donc avoir jusqu'à k processus simultanément en section critique.

Nous allons nous inspirer de l'algorithme proposé en cours (proposé par Ricart & Agrawala en 1983).

Question 1 : Combien d'autorisations faut-il pour recevoir pour garantir qu'au moins une ressource critique est libre ?

Question 2 : Proposer une extension du l'algorithme proposé en cours.

Question 3 : Donner la complexité concernant le nombre de messages pour l'utilisation d'une ressource.

Exclusion mutuelle (algorithme de Carvalho et Roucairol)

Dans l'algorithme vu en cours, le site i demande la permission à j à chaque nouvelle demande pour accéder à la ressource critique. Nous allons considérer l'amélioration suivante : puisque j a donné sa permission à i, i la considère comme acquise jusqu'à ce que j demande à i sa permission.

Question : Donner une façon d'implémenter le principe : "le noeud i a la permission du noeud j".

Question : Montrer que la propriété de sûreté est garantie.

Question : Montrer que tout site voulant accéder à la ressource critique y accédera (pas de phénomène de famine).

Question : Décrire complètement l'algorithme.

Question : Donner une exécution où 3 sites veulent accéder à la ressource critique avec un site qui réclame deux fois de suite une ressource.

Amélioration de l'algorithme (Algorithme de Maekawa)

Dans l'algorithme vu en cours, à chaque nouvelle demande pour accéder à la ressource critique, le site va demander la permission à tous les autres. Au lieu de demander à tout le monde, chaque site i a un ensemble de sites RS_i à qui il va demander la permission d'entrer en section critique. Lorsque i obtient toutes les permissions de RS_i, alors il peut utiliser la ressource critique.

Question : Donner les propriétés sur les différents S_i pourqu'on ait la propriété de sureté.

Question : Afin que tous les sites aient un rôle équitable, on a
- $\forall i \ |RS_i| = K$
- $\forall i, j$, les sites i et j sont dans le même nombre de RS_* que l'on notera D.
- On supposera que $\forall i \ i \in RS_i$

avec n le nombre de sites.

Quelle est la relation entre D et K ?

Quelle est la relation entre n et K ?

Question : Supposons $n = p^2$. Donner une manière de construire les ensembles RS_i.

Question : Dans l'algorithme vu en cours, un site peut donner la permission d'entrer en section critique à plusieurs autres simultanément. Dans cet algorithme, chaque site ne peut donner sa permission qu'à un seul à la fois.

Considérons que trois sites peuvent accèder à la même ressource critique avec $RS_1 = \{1, 2\}$, $RS_2 = \{3, 2\}$, $RS_1 = \{3, 1\}$. Les trois sites 1, 2, 3 demandent à entrer en section critique "simultanément". Que se passe-t-il ?

Question : Que fait-on pour éviter ce problème ? Appliquer votre solution sur le scénario précédent.

Question : Donnez les types de messages échangés

Question : Décrivez l'algorithme.

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

Algorithme utilisant un arbre (Naimi et Théhel)

Dans cet exercice, nous réalisons une exclusion mutuelle en implémentant une file d'attente. Une file va être implémenté en plaçant sur chaque site i un pointeur $suivant_i$ qui indique le successeur de i dans la file d'attente s'il est différent de *null*. Deux problèmes se posent pour l'implémentation de la file.

1. Comment un site sait-il qu'il est en tête de la file ?
2. Comment un site qui n'est pas dans la file peut-il venir se placer en queue de file ?

Question : Proposer un mécanisme qui permet de savoir si un site est en tête de file.

Question : Proposer une structure qui permet de savoir qui se trouve en dernier. Cette structure doit être dynamiquement (puisque chacun des sites qui veut accéder à la ressource critique doit se mettre en fin de file).

Question : Proposer un protocole de gestion de cette structure (en supposant que la file est non-vide).

Question : Proposer une solution simple pour éviter que la file soit vide.

Question : Ecrire la procédure de demande d'accès à la ressource.

Question : Donner un exemple d'exécution où s est le seul site dans la file et que p et q désirent rentrer dans la file. La demande de p est gérée avant celle de q.

Question : Ecrire les procédures d'accès à la ressource, et de libération de la ressource.

Question : Continuer de dérouler l'algorithme en considérant la suite de l'exemple : s libère la ressource et p accède à la ressource. Puis s veut de nouveau accèder à la ressource.

Question : Pourquoi un site demandant la ressource, l'obtiendra-t-il au bout d'un temps fini ?

Question : Quel est le nombre de messages échangés dans le pire des cas lors d'une demande ?