Algorithmique Répartie

Travaux Dirigés (1), Master 1 Informatique

Gestion d'Horloges et Exclusion Mutuelle

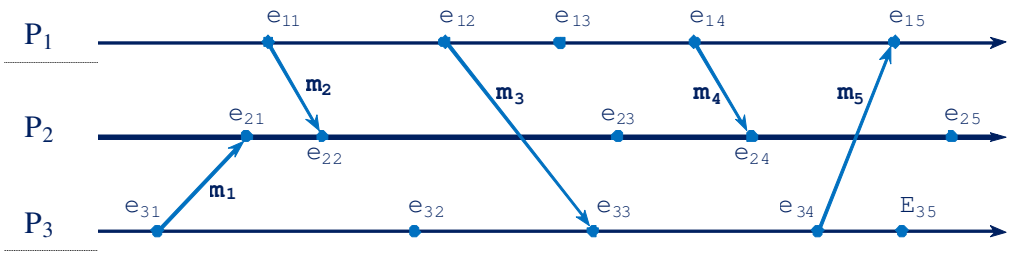
Dans l'ensemble de ce TD, on considère un ensemble de processus indépendants, chaque processus s'exécutant sur un site distinct. Le seul moyen de communication qu'ils possèdent est l'échange de messages. Par la suite, on pourra confondre l'identité d'un processus avec l'identité du site sur lequel il s'exécute.

Hypothèses :

H1 : pas de perte de messages
H2 : pas d'altération de messages
H3 : pas de déséquencement
H4 : réseau maillé
H5 : panne franche de machine

Exercice 1 : Gestion d'Horloges

On considère trois processus coopérants qui réalisent un algorithme quelconque et s'échangent des messages. On s'intéresse à l'exécution suivante :



Toutes les horloges sont initialisées à 0.

1. Donnez les horloges logiques associées à chacun des événements, et les valeurs d'horloge véhiculées par les messages.
2. Donnez les horloges vectorielles associées à chacun des événements, et les valeurs d'horloge véhiculées par les messages.

Exercice 2 : Gestion Centralisée d'une Exclusion Mutuelle Répartie

On considère un ensemble de processus répartis se partageant une ressource critique. Pour garantir l'exclusion mutuelle, on utilise un processus particulier, appelé maître, qui est chargé de gérer les requêtes d'accès à la ressource. Les accès à la ressource se font de manière locale, i.e., le maître ne fait que délivrer les autorisations, il ne traite pas les accès.

1. Proposez un algorithme pour gérer cette exclusion mutuelle (précisez les actions effectuées par le maître, les autres processus et les messages échangés).

2. Représentez sur un exemple impliquant trois sites (P, Q, R) le fonctionnement de cet algorithme. On supposera que les demandes d'entrées en section critique sont reçues par le maître dans l'ordre P, R, Q.
3. Quelle est la complexité en nombre de messages échangés de cet algorithme, lorsque tous les processus demandent une fois la section critique ? Que se passe-t-il si le maître ne respecte pas l'ordre d'arrivée des demandes ?
4. Quelles sont les avantages et inconvénients de cette solution ?
5. Détaillez les actions à effectuer en cas de panne d'une machine (client ou maître).

Exercice 3 : **Algorithme de Lamport (1978)**

L'algorithme de Lamport utilise des horloges logiques pour réaliser une exclusion mutuelle de manière répartie. Dans ce but, les sites s'échangent des messages constitués de 3 champs :

<type du message, horloge locale, n° de site>

Il y a 3 types de messages possibles :

- req : lorsqu'un processus fait une demande d'entrée en section critique ;
- lib : lorsqu'un processus sort de section critique ;
- acq : en réponse à une demande d'entrée en section critique.

Chaque site gère localement une file d'attente de messages. Le principe de l'algorithme est le suivant :

- un processus qui veut entrer en section critique diffuse une requête à tous les autres sites;
- il attend un acquittement de tous ces messages ;
- lorsque sa requête est plus ancienne que les messages reçus de tous les autres sites, il entre en section critique ;
- en sortie de section critique, il diffuse un message à tous les autres sites pour les avertir de la libération de la ressource.

1. Pourquoi inclut-on le numéro de site dans un message ?
2. Est-il nécessaire qu'un site ait reçu les acquittements de tous les autres sites pour entrer en section critique ?
3. Écrire cet algorithme, en précisant les actions effectuées par un processus lors de la réception d'un message (on traitera séparément les différents types de message), lorsqu'il fait une demande d'entrée en section critique.
4. Représenter le déroulement de l'algorithme sur trois sites (P, Q, R)
   - lorsque P est seul à faire une demande,
   - lorsque P et R font une demande en même temps (les requêtes sont envoyées simultanément de P et de R).
   - lorsque P, R, Q font tous une demande.
5. Est-ce que, à tout instant, tous les sites ont la même image de la file d'attente ? Expliquer pourquoi ce n'est pas un problème.
6. Quelle est la complexité, en nombre de messages échangés de cet algorithme ?
7. Détailler les actions à effectuer en cas de panne du site i.

Exercice 4 : Algorithme de Ricart & Agrawala (1981)

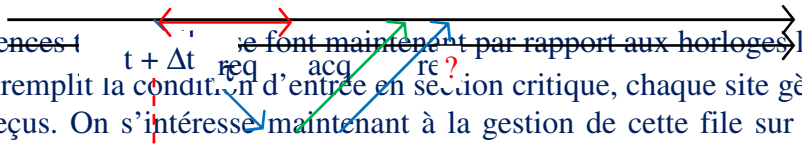
Comme pour l'algorithme de Lamport, on veut maintenant gérer l'exclusion mutuelle de manière complètement répartie : au lieu d'avoir un site maître, chaque site gère localement une copie de la file d'attente. Un processus peut entrer en section critique lorsque sa requête a été acquittée par tous les autres processus.

Un site qui envoie un acquittement autorise donc le processus émetteur de la requête à entrer en section critique. Par conséquent, l'algorithme doit fixer l'instant auquel un site émet un acquittement, pour que la requête servie soit toujours la plus ancienne requête non traitée.

Un site n'émet qu'une requête à la fois. Autrement dit, il ne peut faire une nouvelle demande d'accès en section critique que lorsque la précédente a été satisfaite.

On considère 3 processus i, j et k. i et j ont émis une requête d'entrée en section critique, celle de i est antérieure à celle de j. k n'a pas émis de requête.

1. A quel moment i répond-il à j ? j répond-il à i ? k répond-il à i et j ? Combien de types de messages sont-ils nécessaires pour ces réponses ?
2. Comment un site peut-il être sûr qu'il n'y a pas de requête plus ancienne que la sienne en transit ?
3. Pourquoi n'utilise-t-on pas les horloges physiques pour dater les messages ?



Toutes les références t~~e font maintenant par rapport aux horloges~~ logiques.

Pour savoir s'il remplit la condition d'entrée en section critique, chaque site gère localement une file des messages reçus. On s'intéresse maintenant à la gestion de cette file sur le site i ($File_i$). Les messages stockés dans cette file sont de la forme :

`<type du message, horloge logique, émetteur>`

$File_i$ contient exactement un message en provenance de chaque site.

4. i doit-il systématiquement stocker les acquittements en provenance des autres sites ? Expliquez.
5. Proposez une valeur d'initialisation pour $File_i$.
6. Écrivez cet algorithme, en précisant les actions effectuées par un processus
   - lors de la réception d'un message ;
   - lorsqu'il fait une demande d'entrée en section critique;
   - lorsqu'il sort de section critique.
7. Expliquez pourquoi deux processus ne peuvent pas se trouver simultanément en section critique.
8. À un instant donné, tous les sites ont-ils la même image de la file d'attente ? Expliquez pourquoi ce n'est pas un problème.
9. Quelle est la complexité, en nombre de messages échangés, de cet algorithme pour une demande de section critique ?