Systèmes et algorithmes répartis

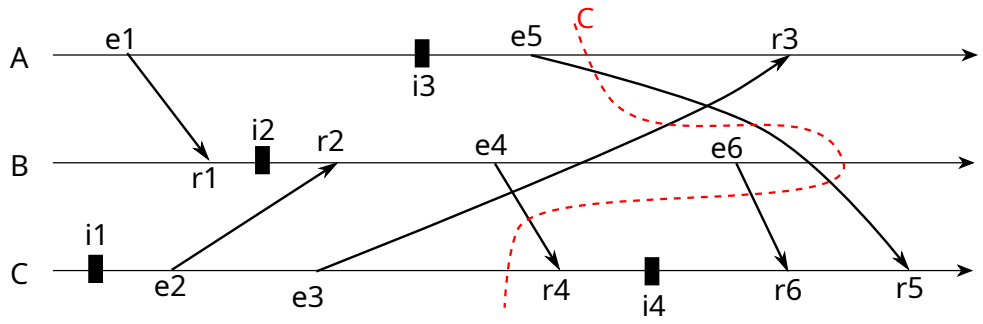
ENSEEIHT/3SN
1h30, documents autorisés

12 décembre 2023

1 point par question, sauf indiqué en marge.

1 Questions de cours (6 points)

On considère les échanges de messages entre 3 sites A, B, C représentés par le chronogramme suivant :



Questions

1. Les événements r_1 et r_3 sont-ils causalement liés ?
2. Donner l'horloge de Lamport de l'événement r_6.
3. Existe-t-il un événement ayant pour horloge vectorielle $(3, 3, 6)$?
4. Donner l'horloge vectorielle de la coupure C.
5. Quelle est l'histoire causale du message reçu en r_6 ?
6. Ajouter un message issu de A pour rendre incohérente la coupure C.

1. non

2. C,8

3. non

4. (3,5,3)

5. m4, m2, m1 (où le message est numéroté comme son événement d'émission)

6. départ avant r_3, réception sur B avant e_6

2 Exclusion mutuelle (8 points)

On considère l'algorithme d'exclusion mutuelle de Ricart-Agrawala, rappelé ci-dessous. L'algorithme suppose que la communication est asynchrone, sans défaillance de réseau, ni défaillance de site. Il garantit qu'au plus un site peut être dans l'état `exclusion` (sûreté) et que tout demandeur finira par obtenir l'accès exclusif (vivacité).

Notes

- Les variables sont locales à chaque site. Quand il est nécessaire de distinguer les variables de deux sites, on précisera en indice le site, par exemple $\texttt{hloc}_i$ et $\texttt{hloc}_j$.
- `hloc` est une horloge de Lamport : `hloc.Top()` incrémente l'horloge locale, `hloc.Recaler(d)` positionne `hloc` à $\max(\texttt{hloc}, \texttt{d}) + 1$.

```
Process P(i : 0..N-1) {
 type Etat = {hors,candidat,exclusion};
 Variables locales
    Etat EC := hors;
    Date hloc := new Date(0,i); // horloge locale
    Date dr; // date de la requête de ce site
    Set<int> Att := ∅;  // sites lui ayant demandé l'autorisation
    Set<int> D;         // sites dont i attend l'autorisation
 on (EC = hors) : // hors → candidat
       EC ← candidat;
       D ← 0..N-1 \ {i};
       dr ← hloc.Top();
       for k ∈ D : send Request(i,dr) to P_k;
 on reception Request(p, d) :
       hloc.Recaler(d);
       if (EC ≠ hors ∧ dr < d) then Att ← Att ∪ {p};
       else send Perm(i) to P_p;
 on reception Perm(p) :  // EC = candidat nécessairement
       D ← D \ {p};
       if (D = ∅) then EC ← exclusion;    // candidat → excl
 on (EC = exclusion) :                    // exclusion → hors
       for k ∈ Att : send Perm(i) to P_k;
       Att ← ∅;
       EC ← hors;
```

Questions

1. Pourquoi faut-il recaler l'horloge du site en recevant un message de requête ?
2. Montrer que l'invariant suivant est vrai :

$$\forall i : EC_i = \texttt{exclusion} \Rightarrow (\forall j : EC_j = \texttt{candidat} \Rightarrow dr_i < dr_j)$$

3. Un étudiant attentif a retenu que les horloges vectorielles sont plus précises que les horloges de Lamport, et il propose de les utiliser pour `hloc` et `dr`. Pourquoi est-ce une mauvaise idée ?
4. Que se passe-t-il si le réseau peut être défaillant et perdre des messages ?
5. Pour corriger cela, on ajoute qu'un site peut renvoyer son message de requête (avec la même date) s'il n'a pas reçu un message de permission au bout d'un certain temps. Montrer que cela invalide l'algorithme tel qu'il est donné en ne garantissant plus l'exclusion mutuelle.
6. Proposer une modification de l'algorithme qui corrige ce défaut.
7. On considère maintenant que le réseau est fiable mais qu'un site peut être défaillant par arrêt définitif. On dispose d'un détecteur de défaillances parfait (P : complétude forte, exactititude forte). Proposer une modification de l'algorithme qui maintient la sûreté (exclusion mutuelle) et la vivacité (pas de blocage).

8. Votre modification est-elle correcte si on utilise le détecteur $\diamond P$ (complétude forte, exactititude finalement forte) ? Que garantit alors l'algorithme ?

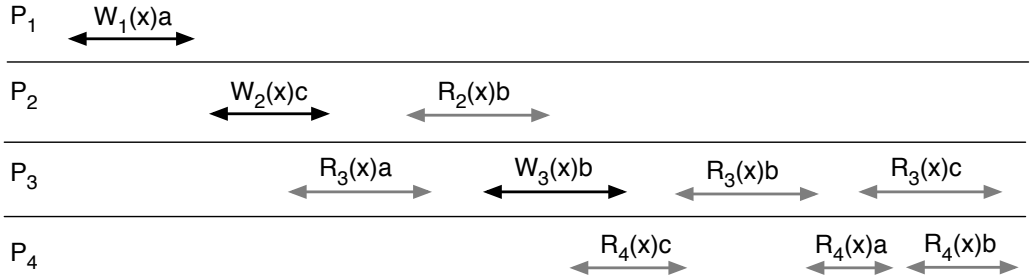
1. *Principe des horloges de Lamport : causalité $\Rightarrow$ temps*
2. *un site ne répond que s'il est hors ou (candidat et plus récent)*
3. *On perd l'ordre total, comment ordonner deux requêtes indépendantes ?*
4. *interblocage : un site dont la requête ou la permission a été perdue ne pourra jamais entrer en exclusion et bloquera toutes les demandes postérieures à la sienne.*
5. *un site peut alors renvoyer plusieurs Perm, un seul sera consommé pour entrer en excl et les suivants (supposés lents) pourraient servir à satisfaire une demande ultérieure.*
6. *mettre la date de la requête dans le message de permission pour distinguer une retransmission d'un nouveau message.*
7. *Un site suspecté est retiré de D. Comme le détecteur est parfait, tout site défaillant finira par être enlevé et on n'enlève pas à tort un site.*
8. *Probablement pas : pendant un certain temps, $\diamond P$ se trompe en suspectant à tort des processus corrects dont on n'attendra pas la permission. L'algo ne garantit que l'éventuelle exclusion mutuelle.*

3 Données réparties (4 points)

Pour les questions qui suivent, justifiez chacune de vos réponses : **seule la justification sera notée**. En ce qui concerne les questions 1 à 3, la justification pourra consister en un argument suffisamment précis (si la réponse est affirmative) ou en un contre-exemple précis (si la réponse est négative).

1. Une exécution linéarisable est-elle nécessairement séquentiellement cohérente ? (0,5 point)
2. Une exécution séquentiellement cohérente est-elle nécessairement causalement cohérente ?
3. Une exécution causalement cohérente vérifie-elle nécessairement la cohérence FIFO ? (0,5 point)

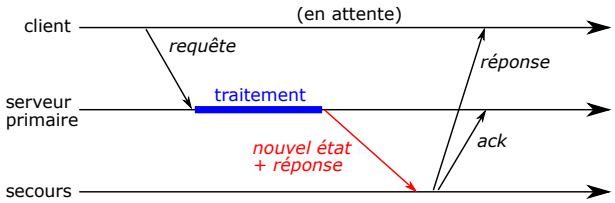
On considère le scénario suivant d'accès aux copies d'une variable x valant initialement 'k', et dupliquée sur les sites P_1, P_2, P_3 et P_4. Chaque opération est accompagnée d'une double flèche, placée en dessous du texte de l'opération, qui représente la plage d'exécution de l'opération.



4. Cette exécution est-elle linéarisable ? Vérifie-t-elle la cohérence séquentielle ?
5. Vérifie-t-elle la cohérence causale ? La cohérence FIFO ?

4 Tolérance aux fautes (2 points)

Le schéma d'Alsberg & Day propose une amélioration de la technique de tolérance aux fautes par redondance passive, dans le cas de 2 serveurs. Cette amélioration consiste à envoyer la réponse au client depuis le serveur de secours, en même temps que l'envoi de l'acquittement (ack) au primaire, au lieu d'envoyer cette réponse depuis le serveur primaire, après qu'il ait reçu l'acquittement du serveurs de secours. Le client peut ainsi obtenir la réponse à sa requête plus rapidement.



1. Pourquoi peut-on anticiper l'envoi de la réponse dans le cas du schéma d'Alsberg & Day ?
2. L'envoi de l'acquittement reste-il nécessaire dans ce cas ? (justifiez votre réponse)