Systèmes et algorithmique répartis

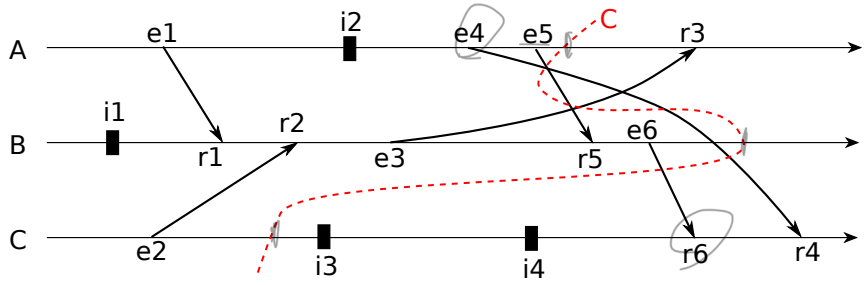
ENSEEIHT/3SN
1h30, documents autorisés

15 décembre 2020

1 point par question, sauf indiqué en marge. Total : 21 points.

1 Questions de cours (7 points)

On considère le calcul réparti suivant avec trois processus :



1. Quel est le passé causal de r_5 ?
2. e_4 et r_6 sont-ils causalement liées ?
3. Quelle est la plus longue chaîne causale conduisant à r_3 ?
4. Donner l'horloge de Lamport de r_5.
5. Donner l'horloge vectorielle de e_6.
6. La coupure C est-elle cohérente ? (justifier la réponse)
7. En notant m_i le message émis en e_i et reçu en r_i, donner l'histoire causale du message m_6.

2 Généralisation de l'algorithme de Ricart et Agrawala (4 points)

On considère l'algorithme de Ricart et Agrawala pour l'exclusion mutuelle avec N sites. Pour obtenir l'accès exclusif, un site intéressé envoie une requête à tous les autres sites et attend d'avoir reçus $N-1$ autorisations. La requête est datée (datation de Lamport) et un site donne son autorisation s'il n'est pas en attente de l'accès exclusif ou si sa propre requête est postérieure. Quand un site libère l'accès exclusif, il envoie une autorisation à toutes les requêtes en attente.

On souhaite généraliser cet algorithme pour autoriser jusqu'à L sites en section critique simultanément.

1. Expliquer pourquoi, dans l'algorithme d'origine, il est nécessaire d'utiliser des horloges de Lamport pour ordonner les requêtes, et si cela fonctionnerait avec des horloges vectorielles.
2. Sans vous préoccuper de la sortie, montrer qu'attendre seulement $N-L$ autorisations généralise correctement l'algorithme pour autoriser jusqu'à L sites en critique simultanément.
3. Montrer que l'algorithme est faux si un site qui libère l'accès exclusif envoie une simple autorisation à toutes les requêtes en attente (penser aux autorisations tardives).
4. Proposer une solution pour corriger ce problème.

3 Consensus avec détecteur de défaillance Ω (6 points)

On considère les détecteurs de défaillance Ω et ◇Ω, dit *immediate leader* et *eventual leader*. Quand on l'interroge, le détecteur Ω renvoie un processus correct [1], et le même pour tous les processus.. Le détecteur ◇Ω renvoie aussi un processus correct, mais au début, pas nécessairement le même correct à tous ; il finit par se stabiliser (en temps inconnu) pour se comporter comme Ω : il renvoie alors un même processus correct à tout le monde.

On suppose qu'il y a N sites, dont au plus f défaillants, avec $f < \frac{N}{4}$. Des étudiants bien intentionnés proposent les algorithmes suivants pour réaliser le consensus. Pour Ω, il suffit d'une itération et tout le monde choisit la valeur du leader. Pour ◇Ω, l'idée est d'itérer jusqu'à ce que suffisamment de sites soient d'accord sur l'unicité d'un leader et choisissent alors sa valeur.

3.1 Consensus avec Ω

```
Processus P_i(v_i),  0 ≤ i < N    -- v_i = valeur proposée par i
   -- leader et v sont locales à P_i
   leader ← Ω   -- interroger le détecteur de défaillance
   envoyer Requête à leader
   attendre Proposition(v)
   décider v et fin
on Requête from q :
      répondre Proposition(v_i) à q
```

1. Cet algorithme vérifie-t-il l'accord ? (toutes les réponses doivent être argumentées)
2. Cet algorithme vérifie-t-il la validité ?

3.2 Consensus avec ◇Ω

```
Processus P_i(v_i),  0 ≤ i < N    -- v_i = valeur proposée par i
    -- leader et v sont locales à P_i
   boucle
      leader ← ◇Ω   -- interroger le détecteur de défaillance
      envoyer Requête à leader
      attendre Proposition(v)
      envoyer Unicité(leader) à tous
      attendre N − f réponses
      si plus de 3N/4 Ack, alors
          v_i ← v   -- mémoriser la valeur décidée
          décider v et fin
      finsi
   on Requête from q :
      répondre Proposition(v_i) à q
   on Unicité(l) from q :
      si l = leader alors répondre Ack à q, sinon répondre Nack à q.
```

3. Cet algorithme vérifie-t-il l'accord ? Imaginer d'abord une exécution sans aucune faute. **(2 pt)**
4. Cet algorithme vérifie-t-il la validité ?
5. Cet algorithme vérifie-t-il la terminaison ?

1. Noter que les détecteurs de défaillance vus en cours renvoient un ensemble de sites suspects ; ici Ω et ◇Ω renvoient un site correct.

4 Réplication de données (6 points)

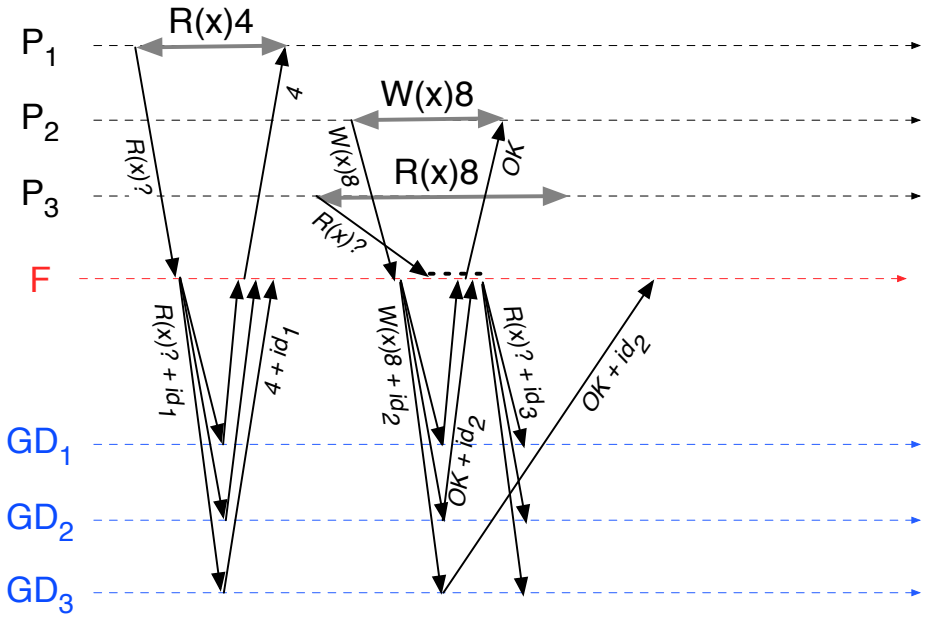
On considère un service réparti de données partagées basé sur la réplication, possédant les caractéristiques suivantes :

- Aucun client ne dispose d'une copie locale. Il passe par un service d'accès aux données partages qui met en œuvre la réplication, de manière transparente pour le client, en gérant les différentes copies sur différents serveurs.
- Le réseau est supposé fiable et asynchrone, et tous les sites sont fiables pour les deux premières questions.

Plus précisément, l'architecture et le protocole sont les suivants :

- L'interaction entre les clients et le service se fait via un site/processus frontal F.
- Le client envoie sa requête (qui est soit une opération de lecture, soit une opération d'écriture d'une donnée partagée), et attend la réponse du frontal F.
- Les copies sont répliquées sur différents gérants de duplication (GD), chaque gérant ayant une copie :
  - Le frontal F ajoute un identifiant unique à la requête du client. Il diffuse ensuite, par un service de **diffusion atomique**, la requête du client (et l'identifiant) aux différents gérants de duplication. Il attend ensuite la réponse des gérants de duplication.
  - **Chaque** gérant de duplication traite la requête, puis renvoie le résultat, accompagné de l'identifiant de la requête.
  - Le frontal F renvoie au client le premier résultat dont l'identifiant correspond à la requête transmis aux gérants de duplication (les autres résultats seront ignorés).
  - Le frontal F peut alors passer à la requête suivante.

La figure suivante illustre l'exécution de ce protocole pour trois clients (P_1, P_2, P_3) et trois gérants de duplication (GD_1, GD_2, GD_3). Le retour de la requête du client P_3 n'est pas représenté, pour des raisons de lisibilité.



1. Ce protocole assure-t-il la cohérence séquentielle ? Justifier la réponse, en donnant un contre-exemple dans le cas où elle est négative, ou en donnant des arguments dans le cas où elle est positive. **(1,5 pt)**

2. Ce protocole assure-t-il la linéarisabilté ? Justifier la réponse, en donnant un contre-exemple dans le cas où elle est négative, ou en donnant des arguments dans le cas où elle est positive. **(1,5 pt)**
3. On considère les défaillances d'arrêt des gérants de duplication (le frontal est supposé fiable). Si l'on dispose de N gérants de duplication, quel est le nombre maximal de défaillances d'arrêt tolérées par ce protocole ? **(1,5 pt)**
4. On souhaite prendre en compte des défaillances byzantines des gérants de duplication (le frontal est supposé fiable). Proposer une adaptation de ce protocole qui permettrait de prendre en compte ce type de défaillance, et donner le nombre maximal de défaillances tolérées, en supposant que l'on dispose de N gérants de duplication. **(1,5 pt)**