Systèmes et algorithmique répartis

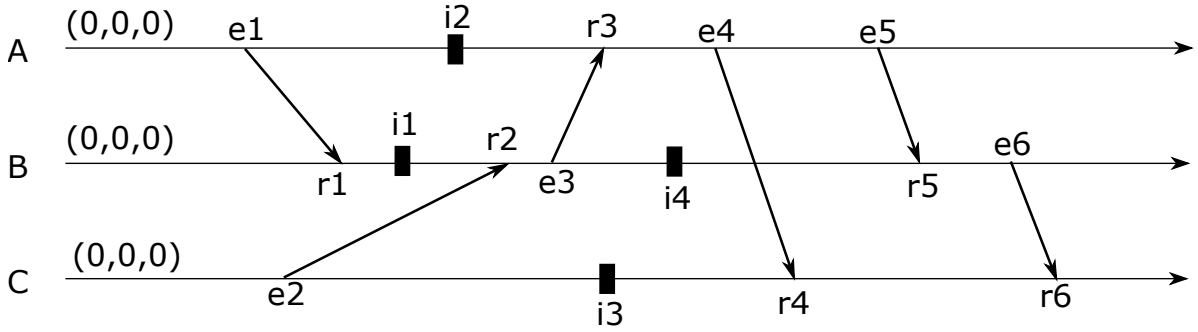
ENSEEIHT/DIMA, master 2 Informatique et Télécommunication
1h45, documents autorisés

10 décembre 2014

1 point par question, sauf indiqué en marge.

1 Calcul réparti et causalité (6 points)

On considère la figure suivante décrivant une exécution d'un calcul réparti qui comporte 3 processus :



Questions (1 point par question)

1. Ce calcul est-il un calcul diffusant ?
   Non : plusieurs événements émission au départ + émission spontanée e_5
   Note : un calcul diffusant n'est pas un calcul où les communications sont des diffusions !
2. Quel est le passé causal de e_5 ?
   $\{e_4, r_3, i_2, e_1, e_3, r_2, i_1, r_1, e_2\}$
3. Quelle est la plus longue chaîne causale conduisant de i_1 à r_6 ?
   $i_1 \rightarrow r_2 \rightarrow e_3 \rightarrow r_3 \rightarrow e_4 \rightarrow e_5 \rightarrow r_5 \rightarrow e_6 \rightarrow r_6$
4. Compléter la valeur des horloges vectorielles pour tous les événements.
   A : $e_1 = (1,0,0)$ $i_2 = (2,0,0)$ $r_3 = (3,4,1)$ $e_4 = (4,4,1)$ $e_5 = (5,4,1)$
   B : $r_1 = (1,1,0)$ $i_1 = (1,2,0)$ $r_2 = (1,3,1)$ $e_3 = (1,4,1)$ $i_4 = (1,5,1)$ $r_5 = (5,6,1)$ $e_6 = (5,7,1)$
   C : $e_2 = (0,0,1)$ $i_3 = (0,0,2)$ $r_4 = (4,4,3)$ $r_6 = (5,7,4)$
5. En utilisant les horloges vectorielles, indiquer si r_4 et i_1 sont causalement liés. Idem pour r_5 et i_3.
   $r_4 = (4,4,3) > i_1 = (1,2,0)$
   $r_5 = (5,6,1) \;||\; i_3 = (0,0,2)$

6. On considère un événement x et son horloge vectorielle H. La somme des valeurs des éléments du vecteur est-elle égale à la longueur du plus long chemin causal menant à x ?
   Non, par exemple r_5

2 Problème : diffusion totalement ordonnée (14 points)

L'objectif du problème est d'étudier la réalisation d'un algorithme de diffusion fiable totalement ordonnée (aussi nommée diffusion atomique) dans un système ne possédant que de la communication point-à-point. La diffusion totalement ordonnée vérifie les propriétés suivantes :

- Validité : si un processus délivre un message m, alors un processus avait diffusé ce message m ;
- Intégrité : aucun message n'est délivré plus d'une fois ;
- Terminaison : si un processus diffuse un message m, alors chaque processus correct (y compris l'émetteur du message) finira par délivrer m ;
- Ordre total : si un processus délivre m avant m', alors aucun processus correct ne délivre m' avant m.

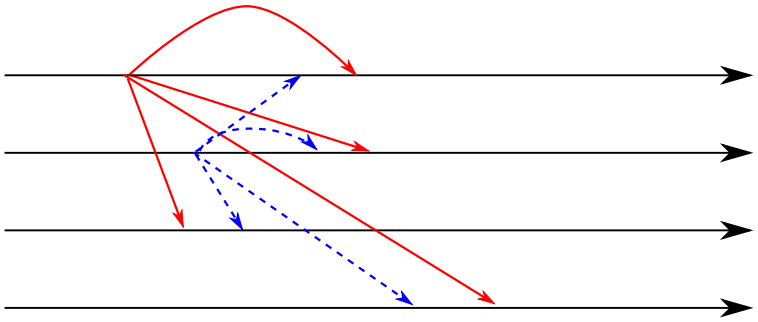


FIGURE 1 – Deux diffusions non totalement ordonnées

Le système Le système est composé d'un ensemble fixe de processus $\{p_i \mid i \in 0..N-1\}$ qui ne communiquent qu'en point-à-point. Les opérations élémentaires pour un processus p_i sont l'envoi d'un message à un processus p_j (noté $send(m)$ *to* p_j) et l'événement de réception d'un message (noté *on* $receive(m)$). L'objectif est de réaliser une opération $to\_broadcast(m)$ (totally-ordered broadcast) qui diffuse un message à tous les processus (y compris soi-même) et l'opération de délivrance $to\_deliver(m)$ d'un message garantissant les propriétés de la diffusion totalement ordonnée.

Hypothèses Sauf explicitement indiqué, on considère que le système est asynchrone (pas de borne temporelle sur le temps de transfert des messages ni sur le temps de réaction d'un processus), que le réseau est fiable (pas de perte ni de duplication de messages) mais sans ordre (tout message arrive éventuellement à destination mais un message « rapide » peut doubler un message « lent »), et que les processus sont fiables (pas de pannes).

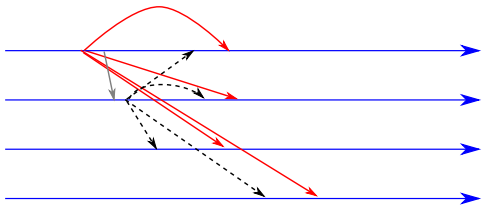
Questions

1. Quand un processus diffuse un message totalement ordonné, il se l'envoie aussi à lui-même. Montrer que, sauf cas particulier, il est impossible qu'il se délivre immédiatement ce message (un dessin est sans doute la meilleure base pour l'explication).

   Cf figure 1 : si les sites 1 et 2 se le délivrent immédiatement, les messages rouge et bleu seront délivrés en ordre inverse sur ces sites.

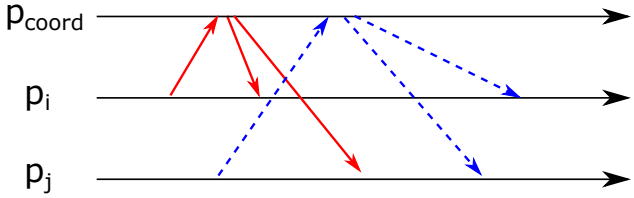
2. Montrer qu'il est parfaitement possible d'avoir une diffusion totalement ordonnée qui ne soit pas une diffusion causale (i.e. dont l'ordre de délivrance respecte l'ordre causal des émissions), en particulier s'il existe des messages point-à-point (de nouveau, un dessin est sans doute la meilleure illustration).

   À cause du message gris, la diffusion noire est causalement postérieure à la diffusion rouge mais la délivrance respecte l'ordre totale (tous les sites délivrent noir avant rouge).
   Note : réponse erronée sans le message gris : les deux diffusions ne seraient pas causalement liées.



2.1 Réalisation avec un processus de coordination

On ajoute un processus de coordination p_{coord}. Quand un processus souhaite diffuser un message à $p_0, \ldots, p_{N-1}$, il l'envoie au processus de coordination. Le processus de coordination l'envoie alors à tous les processus.



Les algorithmes sont :

```
-- Processus p_i
to_broadcast(m) :
  send m to p_coord

on receive(m) :
  to_deliver(m)
```

```
-- Processus p_coord
on receive(m) :
  for i = 0 to N-1 :
      send m to p_i
```

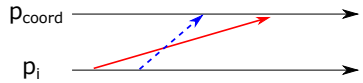
Questions

3. Justifier que cette implantation est correcte sous l'hypothèse que les canaux sont fifo.
   Rappel : la communication est fifo ssi, soit m un message de p_j à p_k, et soit m' un message de p_j à p_k émis après m, alors m est reçu avant m' (même émetteur, même destinataire).

   Les trois propriétés Validité, Intégrité, Terminaison sont trivialement vraies. Pour l'ordre total : tout passe par le coordinateur ⇒ que des messages p_i vers coord, ou coord vers p_i. Si deux diffusions consécutives sont effectuées par le coordinateur, p_i quelconque recevra les

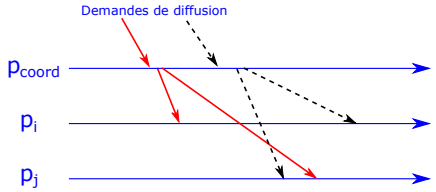
deux messages nécessairement dans l'ordre d'émission du coordinateur (hypothèse fifo) et donc l'ordre total est garanti.

4. La communication n'est pas fifo : le coordinateur peut recevoir dans l'ordre inverse deux demandes consécutives de diffusion venant d'un même p_i :



Montrer que l'algorithme ne garantit pas non plus que tous les destinataires délivreront les messages dans le même ordre (propriété ordre total).

Deux diffusions m et m', transmises de coord vers p_i et p_j, peuvent être reçus en sens différent sur p_i et p_j, invalidant l'ordre total.
Note : l'origine des demandes de diffusions n'a pas d'importance.



5. Introduire des compteurs pour rendre correct l'algorithme. Indiquer soigneusement l'information associée aux messages. **2pt**

Au choix :

(a) Implanter comme un bourrin des canaux fifos ;

(b) Un numéro global de séquence sur le coordinateur + un compteur sur chaque site pour le prochain message attendu du. Note : ce n'est pas important si 2 messages consécutifs issus d'un même site vers le coordinateur se croisent (ça respecte l'ordre total). TODO : détailler la solution.

6. L'algorithme garantit-il que la délivrance est causalement ordonnée ? Justifier.

Non : le coordinateur ne fait pas de lien entre deux demandes de diffusion : si une demande de p_j lui parvient avant une demande causalement *antérieure de p_i (ce qui est parfaitement possible même si la communication est fifo : pas même émetteur), la demande de p_j sera traitée avant celle de p_i. TODO : faire un schéma.*

7. Dans cette question, on considère qu'un processus peut être défaillant par arrêt définitif. L'algorithme est-il toujours correct ? Si oui, justifier ; si non, quelle(s) propriété(s) de la diffusion totalement ordonnée est(sont) invalidée(s) et pourquoi.

Non, si c'est le coordinateur qui s'arrête pendant sa boucle d'émission. Propriété terminaison invalidée. Par contre, pas de problème si un autre processus s'arrête (la propriété terminaison porte sur les processus corrects).

8. Dans un système où les seules communications sont celles de la diffusion totalement ordonnée, montrer que dans toute coupure cohérente, le nombre de messages reçus par le coordinateur est nécessairement supérieur ou égal au nombre de messages reçus par n'importe quel autre site.

Si un site p_i a reçu un message m et cette réception est dans la coupure, ce message provient d'une émission par le coordinateur, qui l'avait faite car ayant reçu un message de demande m'. Vu que $m' \prec m$, m' est dans la coupure. Par ailleurs, le coordinateur n'envoie qu'un message à chaque site pour chaque demande reçue. Donc pour tout message reçu par un site quelconque et qui est dans la coupure, il y a un message reçu par le coordinateur qui est dans la coupure.
Autrement dit, réception sur $p_i \succ$ émission sur coord $\succ$ réception sur coord.

2.2 Réalisation avec un jeton circulant

Au lieu d'avoir un processus externe qui sert de coordinateur, nous allons utiliser un jeton circulant. Quand un processus veut diffuser un message, il attend d'avoir le jeton. Pour que le jeton circule sur un anneau logique constitué par les processus dans l'ordre de leur identité, on prend $succ(i) = (i+1)\%N$.

```
-- Processus p_i
to_broadcast(m) :
  wait for jeton_i = true
  for i = 0 to N-1 :
      send m to p_i
  send token to p_succ(i)   -- envoi au processus suivant
  jeton_i := false

on receive(m) :                          on receive(token) :
  to_deliver(m)                            jeton_i := true

on jeton_i = true :   -- pas besoin du jeton : transmission au suivant
  send token to p_succ(i)
  jeton_i := false
```

Questions

9. Montrer que cette implantation est incorrecte même si les canaux sont fifo. Est-elle correcte si les canaux respectent la causalité ?

   Jeton qui irait plus vite sur son anneau qu'un message direct : p_i diffuse un message m en particulier vers p_k et transmet le jeton à p_j ; p_j diffuse m', en particulier vers p_k. p_k peut recevoir m' avant m.

   Par contre, l'algorithme est correct si les canaux respectent la causalité : diffusion de m par $p_i \prec$ envoi du jeton à $p_j \prec$ diffusion de m' par p_j, donc m' est causalement postérieur à m et ne peut être délivré avant.

10. Les canaux ne sont pas fifo. Comme précédemment, introduire des compteurs pour rendre correct l'algorithme. Bien préciser les informations transmises par le jeton et les messages.

    (a) Très très bourrin, implanter la causalité (il faut les matrices d'horloges)

    (b) Un numéro de séquence, transmis par le jeton, incrémenté et associé aux messages : le site qui a le jeton associe le numéro de séquence courant aux messages qu'il envoie + il incrémente le numéro de séquence transmis avec le jeton au site suivant. Un site délivre un message si le numéro associé est le numéro qu'il attend (il incrémente alors ce numéro), sinon il met le message de côté en attente du ou des messages en retard.

2.3 Jeton circulant avec défaillance d'arrêt

On considère maintenant des défaillances de processus de type arrêt définitif. On ne se préoccupe pas de la structure de l'anneau : on suppose que $succ(i)$ est toujours le processus correct suivant

pour maintenir un anneau (si $succ(i) = j$ et j s'arrête, $succ(i)$ devient $j+1$ ou $j+2$, etc. L'anneau est maintenu tant qu'il reste au moins un processus).

Questions

11. Montrer que le problème de la diffusion totalement ordonnée avec défaillance d'arrêt en asynchrone est impossible à résoudre sans détecteur de faute. Pour cela, il suffit de donner une implantation du consensus s'appuyant sur la diffusion totalement ordonnée : si la diffusion totalement ordonnée était possible, alors le consensus serait aussi possible. Or on sait que le consensus est impossible en asynchrone avec défaillance d'arrêt.

    Tout le monde diffuse sa valeur. La première valeur reçue gagne. Ordre total ⇒ tout le monde reçoit toutes les valeurs dans le même ordre. On a donc les quatre propriétés du consensus : accord (ordre total = tout le monde décide pareil), intégrité (un seul point de décision), validité (une des valeurs diffusées), terminaison (car la diffusion termine). Donc, si on avait une implantation de l'ordre total, on aurait une implantation du consensus. Vu qu'on sait que le consensus est impossible avec défaillance d'arrêt, c'est que l'ordre total est impossible aussi.

12. On suppose qu'on dispose d'un détecteur de fautes de type W (complétude faible, exactitude faible). Donner le principe d'une solution permettant de détecter la disparition du jeton quand le site qui le possède s'arrête, et de le recréer sur un autre site correct. On pourra s'appuyer sur les algorithmes vus en cours.

    Consensus pour détecter la perte du jeton + élection pour choisir qui le recrée.

2.4 Diffusion totalement ordonnée et P2P structuré

13. Expliquer sommairement (3 ou 4 phrases) quels sont les points difficiles pour réaliser une diffusion totalement ordonnée dans un système P2P basé sur une table de hachage distribuée (DHT).

    - *Routage multi-hop où personne ne connaît toute la table*
    - *Nombre de sites a priori très grand : ordre total ⇒ accord global.*
    - *Appartenance dynamique : il faudrait ordonner totalement les arrivées/départs de sites par rapport aux diffusions*