Corrigé

Examen d'algorithmique distribuée

EPITA ING1 2012 S2; A. DURET-LUTZ

Durée : 1 heure 30

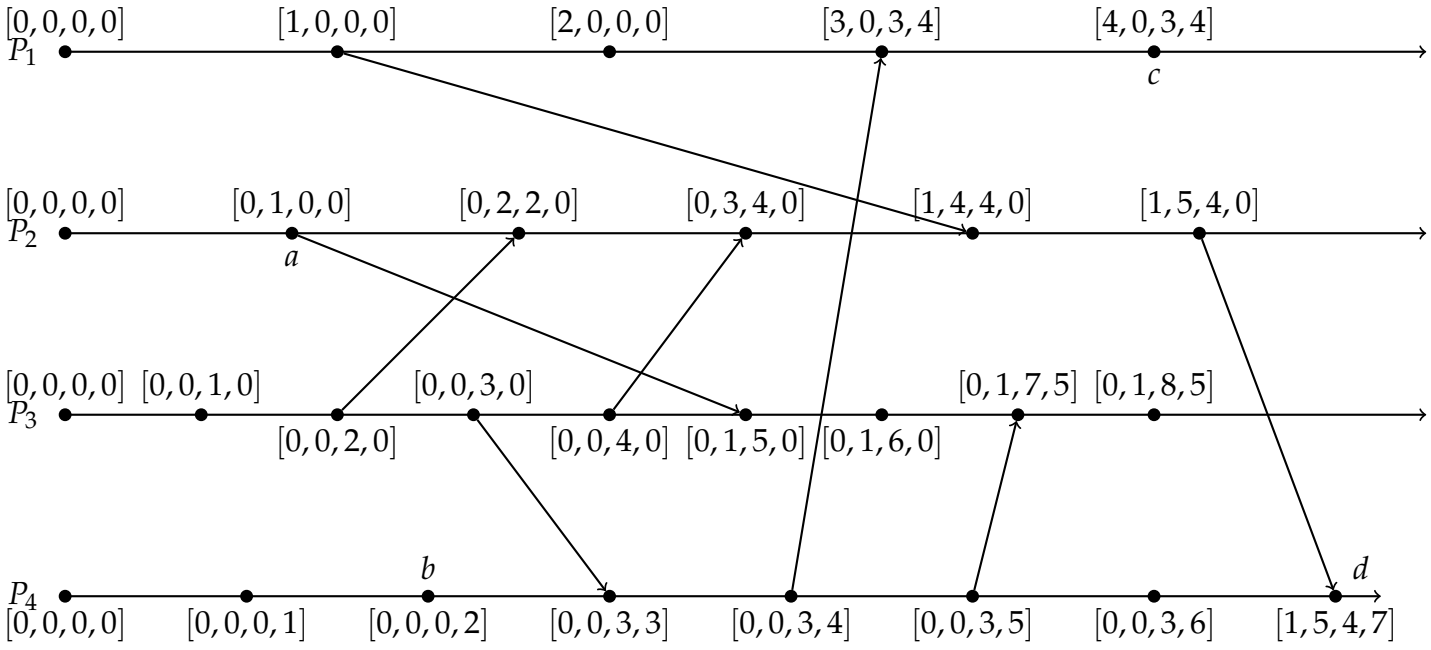
Juin 2010

Consignes

- Tous les documents (sur papier) sont autorisés.
  Les calculatrices, téléphones, PSP et autres engins électroniques ne le sont pas.
- Répondez sur le sujet dans les cadres prévus à cet effet, ainsi que sur les figures.
- Il y a 5 pages d'énoncé.
  Rappelez votre nom en haut de chaque feuille au cas où elles se mélangeraient.
- Le barème est indicatif et correspond à une note sur 22.
- **Sauf indication contraire, les canaux sont supposés FIFO et sans perte.**

1 Horloges vectorielles et causalité (5 points)

On considère un système distribué constitué de 4 sites P_1, P_2, P_3, P_4, s'envoyant des messages de façon asynchrone comme représenté par la figure suivante. Les événements d'un processus, représentés par des gros points noirs, sont soit des événements locaux (étapes d'un calcul), soit des envois ou des réceptions de messages. Ces événements sont datés par un système d'horloges vectorielles, initialisées à $[0,0,0,0]$ dans chaque état initial.



1. **(3 points)** Indiquez à côté de chaque point de la figure la valeur de l'horloge du processus où se produit l'événement correspondant.

2. **(2 points)** On considère les événements a, b, c et d de la figure. Cochez toutes les formules justes :

$\square\ a \to b$	$\square\ a \to c$	$\boxtimes\ b \to c$	$\boxtimes\ a \to d$
$\square\ b \to a$	$\square\ c \to a$	$\square\ c \to b$	$\square\ d \to a$
$\boxtimes\ a \| b$	$\boxtimes\ a \| c$	$\square\ b \| c$	$\square\ a \| d$

2 Généralisation de l'exclusion mutuelle (5 points)

Une version généralisée du problème de l'exclusion mutuelle consiste à imposer que, sur les N processus du système réparti, seuls $L \geq 1$ processus peuvent être en section critique simultanément. (Dans l'exclusion multuelle normale $L = 1$.) Donc si moins de L processus sont en section critique et qu'un nouveau processus veut lui aussi rentrer en section critique, on doit le lui permettre.

Proposez une modification de l'algorithme de Ricart et Agrawala pour traiter ce problème. Vous considérerez que $N \geq 1$ et $L \geq 1$ sont des constantes fixées au démarrage du système.

Réponse :

On peut imaginer plusieurs approches :

1. Une façon de faire serait de remarquer que dans le cas classique avec un seul processus autorisé en section critique, on y entre lorsqu'on a reçu les autorisations de tout le monde (le processus en section critique ne donnant pas la sienne avant d'en sortir). On attend donc $N - 1$ autorisations.

   Si L processus sont autorisés en section critiques, il suffit d'attendre $N - L$ autorisations. On sait qu'il y aura alors au plus $L - 1$ processus en section critique.

   Lorsque les $L - 1$ processus restant sortent de section critique ils vont envoyer des messages aux processus dont la demande avait été mise en attente, et il est possible que ces messages atteignent des processus qui sont déjà sortis de la section critique dont ils avaient demandé l'entrée. Il est aussi possible que c'est processus aient depuis demandé une nouvelle autorisation. Il est donc important d'installer un mécanisme (par exemple identifier la demande dans l'autorisation) permettant de détecter les autorisations périmées pour les ignorer.

2. Une autre façon de modifier l'algorithme est la suivant : on attend toujours $N - 1$ autorisations, mais les processus qui sont en section critique (ou qui attendent d'y rentrer) peuvent distribuer au maximum $L - 1$ autorisations avant de stocker les demandes dans la file comme dans l'algorithme original.

   Chaque processus P_j qui demande l'entrée en section critique initialise donc un compteur c_j à 1. Ce compteur représente le nombre de processus qui ont demandé l'entrée en section critique après lui et qu'il a autorisé à y rentrer (lui compris).

   Lorsqu'un processus P_i demande l'autorisation d'entrer en section critique à un processus P_j qui est en section critique (ou en attente de section critique), il y a trois possibilités :
   - Soit la demande de P_i précède celle de P_j, auquel cas P_j envoie *grant* à P_i sans modifier c_j.
   - Soit la demande de P_j précède celle de P_i :
     - Si $c_j < L$ alors P_j envoie *grant* à P_i et incrémente c_j.
     - Si $c_j = L$ alors P_j met la demande de P_i dans sa file d'attente comme dans l'algorithme original.

3 Diffusions atomiques avec le protocole ABCAST (12 points)

ABCAST (Birman, 1987) est un protocole permettant de s'assurer que plusieurs diffusions, émises de façon concurrentes, sont délivrées sur tous les sites destinataires dans le même ordre.

Le protocole utilise des horloges de Lamport sur chaque site.

Chaque site possède de plus une file de messages dans laquelle il stocke les diffusions reçues pour pouvoir les réordonner. Les entrées de cette file possèdent quatre champs :
– le contenu du message reçu,
– l'expéditeur du message,
– une estampille indiquant la date de *réception* du message,
– un bit indiquant le statut de cette estampille : soit *provisoire*, soit *définitive*.

Un message n'est délivrable sur un site que si son estampille est définitive et est la plus petite estampille de la file des messages de ce site. Si deux messages de la file ont la même estampille, on les ordonne par rapport à l'expéditeur.

Les règles de l'algorithme ABCAST sont les suivantes :

(A1) Quand un site P_i reçoit une diffusion m de la part de P_j, il l'estampille de façon provisoire avec la valeur h de son horloge locale. Le message m est ensuite placé dans la file d'attente (avec son estampille h et le bit indiquant *provisoire*). De plus l'estampille h est envoyée à P_j. Cette étape fait avancer l'horloge locale du site.

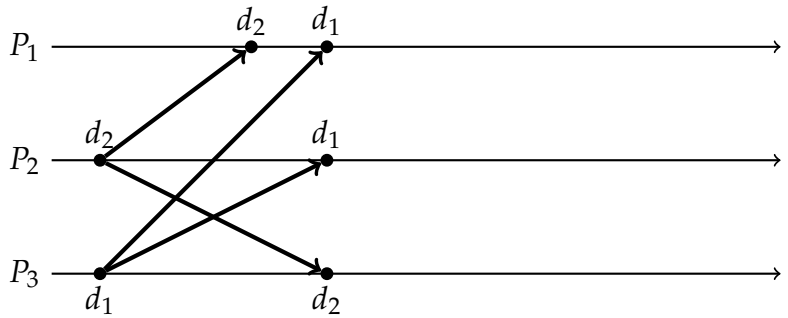
(A2) Quand le diffuseur a reçu les estampilles provisoires proposées par tous les destinataires, il sélectionne la plus grande valeur, puis il la diffuse à tous les destinataire comme estampille finale. Cette étape fait avancer l'horloge locale du diffuseur.

(A3) À la réception de l'estampille finale, chaque site met à jour l'estampille du message correspondant dans la file, en la marquant comme *définitive*. Les sites mettent à jour leur horloge en faisant un max avec l'estampille finale. Il est possible que l'horloge ne change pas lors de cette étape, si l'estampille finale était plus petite que la valeur de l'horloge.

(A4) Juste après l'étape précédente, chaque site s'occupe des messages qui sont délivrables s'il y en a. C'est-à-dire que tant que la plus petite estampille de la file est définitive, on retire le message correspondant de la file, et on le traite. L'horloge avance chaque fois qu'un message est ainsi délivré.

1. **(1 pt)** Oubliez cet algorithme dans un premier temps, et dessinez un scénario dans lequel deux diffusions concurrentes ne sont pas délivrées dans le même ordre.

Réponse :



2. **(2 pts)** Donnez un exemple de situation concrète dans laquelle il est important d'imposer un ordre unique sur les diffusions concurrentes. Autrement dit, donnez une application utile de cet

algorithme.

Réponse :

Supposons une base de donnée répliquée sur plusieurs site. Les transactions effectuées sur un site quelconque sont diffusées aux autres sites.
Si deux sites décident de façon concurrente de faire une mise à jour sur un même enregistrement, ces transactions peuvent être effectuée dans un ordre ou dans l'autre, mais on veut que l'ordre soit le même sur tous les sites.

3. **(5 pts)** La figure qui suit représente une exécution de l'algorithme dans un système possédant huit sites séparés en deux groupes : quatre diffuseurs (P_A, P_B, P_C et P_D) non représentés sur la figure, et quatre cibles (P_1, P_2, P_3 et P_4). Les diffuseurs diffusent chacun un message à destination des cibles. Ces messages, dont l'ordre d'expédition est sans importance, sont désignés respectivement par A, B, C et D.

   Sur la figure, les ronds désignent les étapes **(A1)** de l'algorithme, c'est-à-dire lorsqu'un message de diffusion est reçu par une cible et que cette dernière envoie sa valeur d'horloge au diffuseur. Les carrés désignent les étapes **(A3)**, c'est-à-dire lorsque l'estampille définitive du message est reçue. Enfin les points noirs représentent les moments où les messages de diffusion sont vraiment délivrés et traités (étape **(A4)**). La figure ne montre qu'un point noir sur les 16 qui devraient y apparaître : vous devrez la compléter. Les nombre au dessus des cercles, carrés, ou points, désignent les valeurs des horloges logiques des sites (la encore il vous faudra compléter).

   Pour vous aider, considérez ce qui se passe du point de vue de la diffusion B. P_B diffuse B à toutes les cibles. Celles-ci reçoivent le message aux instants représentés par les B entourés. Les horloges locales des cibles valent alors 7, 6, 9 et 7 respectivement. Ces valeurs sont envoyées à P_B qui prend la plus grande (9), et la diffuse aux cibles. Les cibles reçoivent cette estampille définitive aux instants représentés par des B encadrés et mettent à jour leur file de messages en attente (ces files ne sont pas représentées). Certains sites dont l'horloge étaient inférieure doivent alors mettre à jour leur horloge (c'est le cas de P_2 et P_4 par exemple). Maintenant que l'estampille de B est définitive, ce message pourra être délivré lorsque son estampille sera la plus petite de la file de chaque site (l'instant est désigné par un point noir sur le site P_3).

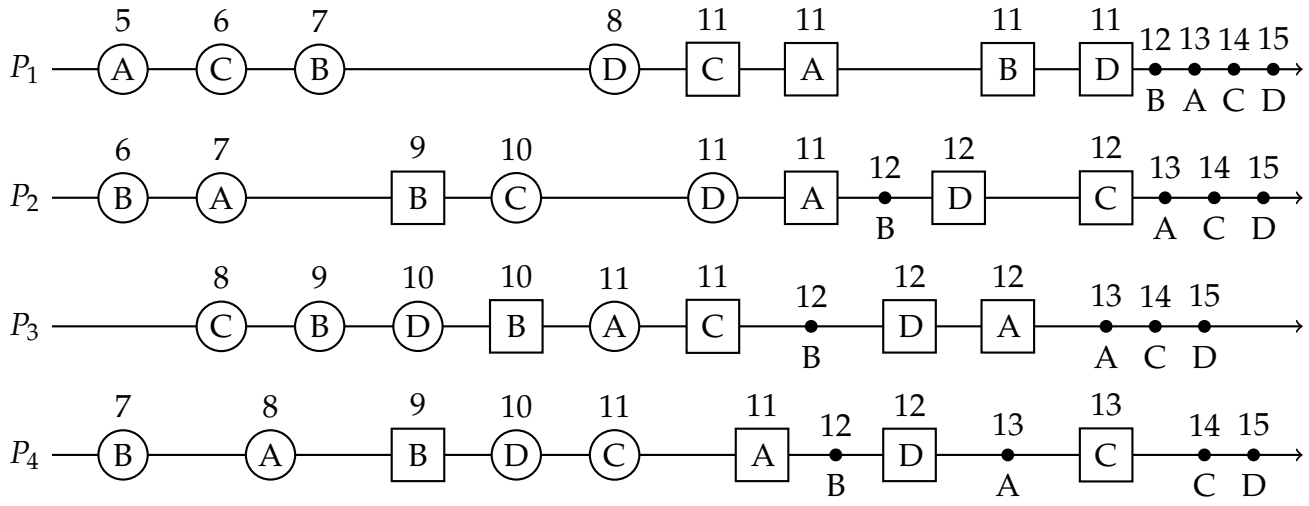
   On vous demande

   (a) d'ajouter les points noirs représentant les instants où sont délivrés les messages sur tous les sites,

   (b) de compléter les valeurs des horloges au dessus de chaque événement (carré, cercle, ou point).

   Il pourra être utile de chercher à répondre à la question suivante avant de terminer celle-ci.

Réponse :



4. **(2 pt)** Dans quel ordre seront finalement délivrés les messages A, B, C et D ? (Pour résoudre d'éventuelles égalités, vous supposerez que l'ordre entre les expéditeurs est $P_A < P_B < P_C < P_D$).

Réponse :

P_A reçoit les estampilles 5, 7, 11 et 8. L'estampille finale de A sera donc 11.
P_B reçoit les estampilles 7, 6, 9 et 7. L'estampille finale de B sera donc 9.
P_C reçoit les estampilles 6, 10, 10 et 11. L'estampille finale de C sera donc 11.
P_D reçoit les estampilles 8, 11, 10 et 10. L'estampille finale de D sera donc 11.

A, C et D ont la même estampille, on utilise donc l'ordre sur les sites pour les départager.

L'ordre final est B, A, C, puis enfin D.

5. **(1 pts)** Cet algorithme reste-t-il correct lorsque les canaux ne sont pas FIFO ? Sinon, comment modifier l'algorithme pour supporter des canaux non-FIFO ?

Réponse :

Oui l'algorithme reste correct. L'algorithme demandant un dialogue entre le diffuseur et les cibles (on attend une réponse avant d'envoyer le message suivant), il n'y a pas de risque d'avoir des messages qui se doublent.

6. **(1 pts)** Cet algorithme reste-t-il correct lorsque les canaux peuvent perdre des messages ? (Justifiez votre réponse.)

Réponse :

Non, l'algorithme ne résiste pas aux pertes de messages : le diffuseur attend une réponse de tous les destinataires. Si l'une de ces réponses est perdue, il attendra indéfiniment.