Corrigé

Examen d'algorithmique distribuée

EPITA ING1 2011 S2; A. DURET-LUTZ

Durée : 1 heure 30

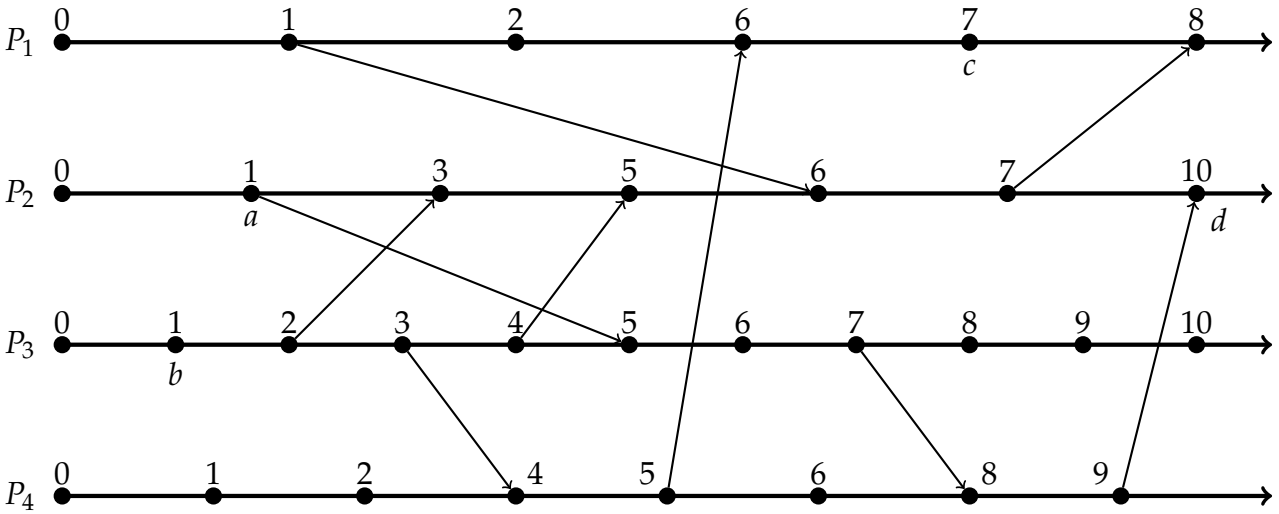
Juin 2009

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 27.

1 Horloges de Lamport et causalité (9 points)

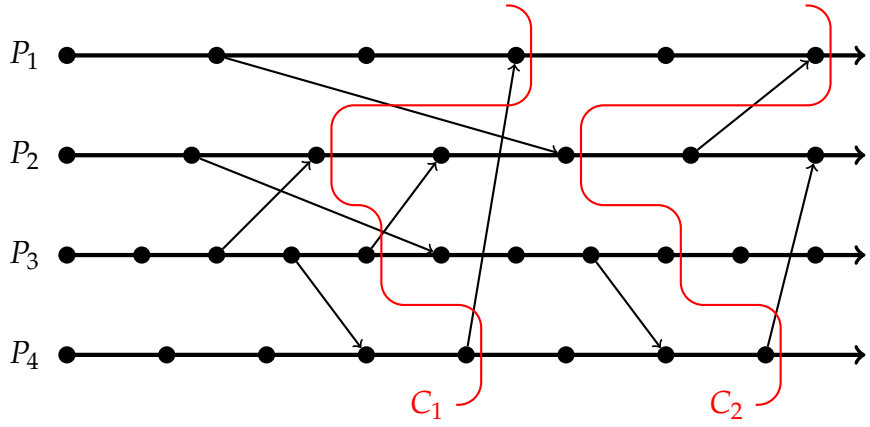
On considère un système distribué constitué de 4 sites P_1, P_2, P_3, P_4, s'envoyant des messages de façon asynchrone comme représenté par la figure suivante. Les événements d'un processus, représentés par des gros points noirs, sont soit des événements locaux (étapes d'un calcul), soit des envois ou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



1. **(2 points)** Indiquez au dessus 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 :

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

On considère les deux coupures désignées C_1 et C_2 dans la figure ci-dessous (qui reproduit les communications de la figure précédente) :



3. **(2 points)** Les coupures C_1 et C_2 sont-elles cohérentes ? Justifiez vos réponses. (N'hésitez pas à nommer des états sur la figure pour vous aider à vous justifier.)

Réponse :

C_1 est cohérente : tous les messages reçus dans la coupures ont étés envoyés dans la coupure.
C_2 n'est pas cohérente : P_1 a reçu un message qui a été envoyé après la coupure.

4. **(3 points)** Supposons que l'état de chaque processus ait été sauvegardé au moment de la coupure. Les valeurs des horloges de Lamport sauvegardées avec chaque processus suffisent-elles pour décider si la combinaison de ces états locaux forme un état global cohérent ? Comment peut-on détecter ces incohérences autrement ?

Réponse :

Les horloges de Lamport ne suffisent pas. Ce n'est pas parce qu'on a $HL(a) < HL(b)$ que $a \rightarrow b$.
Il faudrait utiliser des horloges vectorielles. Dans ce cas la coupure $C = (c_1, c_2, ..., c_n)$ est cohérente ssi pout tout processus P_i on a $HV_i(c_i)[i] = \max_j HV_j(c_j)[i]$.
Ou encore : on pourrait compter, sur chaque site, le nombre de messages émis vers et reçus de tous les autres sites. Pour vérifier qu'une sauvegarde est cohérente il faut alors consulter ces compteurs pour s'assurer qu'un site P_i n'a pas reçu d'un site P_j plus de messages que le site P_j ne lui en a envoyé.

2 Horloges matricielles (9 points)

On considère maintenant un système distribué constitué de trois sites nommés P_1, P_2 et P_3, utilisant des horloges matricielles pour dater les événements de chaque processus.

L'état initial du chaque processus est $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. On considère qu'aucun événement (local ou envoi/réception n'a eu lieu dans l'état initial.

Après quelques instants d'exécution, les horloges des processus P_1, P_2 et P_3 indiquent les dates suivantes :

$$HM_1 = \begin{bmatrix} 3 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} \qquad HM_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad HM_3 = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 3 \end{bmatrix}$$

Pour la suite on supposera que tous les messages envoyés ont été délivrés au plus tard aux dates indiquées. C'est-à-dire qu'aucun message n'est en transit.

1. **(1 point)** Au total, combien de messages ont été échangés ?

Réponse :

Il y a eu quatre message échangés. Les horloges indiquent que P_1 a envoyé un message au processus P_3 ($HM_3[1,3]$), P_2 a envoyé un message à chaque autre processus ($HM_3[2,1]$) et ($HM_3[2,3]$), et P_3 a envoyé un message au processus P_1 ($HM_3[3,1]$).

2. **(1 point)** Au total, combien d'événements ont eu lieu ?

Réponse :

Il y a eu $8 = HM_1[1,1] + HM_2[2,2] + HM_3[3,3]$ événements.

3. **(1 point)** Au total, combien d'événements locaux ont eu lieu ? (c'est-à-dire sans compter les événement correspondants à des émissions ou des réceptions de messages, ni l'état initial)

Réponse :

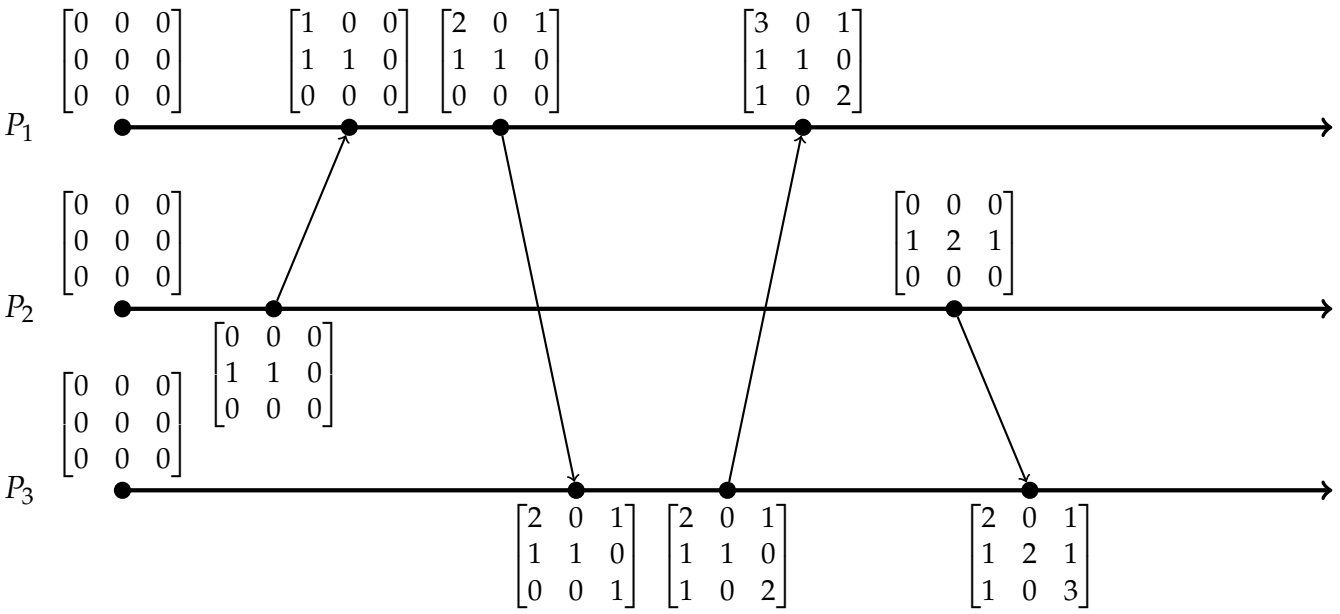
8 événements permettent de faire 4 émissions et 4 réceptions. Comme 4 messages ont étés échangés, il n'y a eu aucun événement local.

4. **(2 points)** Dans quel ordre ont été envoyés les messages de P_2 ? (Justifiez votre réponse.)

Réponse :

P_2 a envoyé deux messages : l'un vers P_1 et l'un vers P_3. D'autre part $HM_2[2,2]$ nous dit qu'il n'a exécuté que deux événements. S'il avait envoyé le message vers P_1 en deuxième on aurait (121) comme seconde ligne de HM_1.
Le message à P_1 a donc été envoyé avant le message à P_2.

5. **(4 points)** Complétez le schéma suivant pour reconstruire l'historique des événements jusqu'aux dates indiquées. Vous indiquerez la date (matricielle) de chaque événement.



3 Un Parking (9 points)

Un parking possède p portes par lesquelles des voitures peuvent entrer ou sortir. Le parking ne doit jamais contenir plus de N voitures. À chaque porte se trouve un gardien ; les gardiens communiquent

en s'envoyant des messages (de façon asynchrone, par exemple à l'aide de pigeons voyageurs supposés fiables) ; ils doivent faire en sorte que la capacité du parking ne soit jamais dépassée. Il faut par ailleurs que, s'il y a régulièrement des sorties, toute voiture qui attend pour entrer *par la porte de son choix* y parvienne en un temps fini.

Décrire un algorithme réparti, qui sera appliqué par chacun des gardiens, afin résoudre le problème. Cet algorithme utilisera des principes présentés en cours. Le nombre de messages échangés entre deux entrées de voitures devra être borné.

Mises en gardes :

- Une idée (incorrecte) serait de partager initialement les N places du parking en p lots de N/p places, à la responsabilité de chaque gardien. Dans ce cas le gardien peut laisser rentrer N/p voitures, plus autant de voitures qu'il a laissé sortir. Cependant, avec ce schéma si toutes les voiture sortent par la même porte, cette dernière porte devient la seule entrée possible et les voitures qui cherchent à rentrer par une autre porte y restent bloquées (ce qu'on ne veut pas)
- Une autre idée (tout aussi incorrecte) serait d'imaginer que les gardiens tentent de garder le compte des places libres en diffusant (aux autres gardiens) des messages "entrée" ou "sortie" à chaque fois qu'une voiture entre ou sort par leur porte. Le problème est que ces diffusions ne sont pas délivrées instantanément et qu'elles peuvent aussi se croiser. Cela devient critique lorsqu'il n'y a plus beaucoup de places libres. Par exemple s'il ne reste plus qu'une place libre, deux gardiens pourraient simultanément laisser rentrer chacun une voiture.

Réponse :

On peut utiliser une variante des algorithmes d'exclusion mutuelle à jetons vus en cours. On stocke sur le jeton le nombre de places disponibles (en plus des informations nécessaires au protocole de d'exclusion). Un gardien ne peut laisser rentrer ou sortir une voiture que s'il a le jeton et qu'il peut mettre à jour le nombre de places libres.
D'autre part si un gardien qui souhaite faire rentrer une voiture récupère un jeton indiquant qu'aucune place n'est libre, il ne fera pas rentrer la voiture et redemandera le jeton aussitôt qu'il l'aura cédé à un autre gardien.