**Algorithmes Distribués TD 4 : Etat Global et Snapshot**

**Exercice 1**

Considérons un système composé de trois processus numérotés 0, 1 et 2 connectés par des canaux FIFO (figure ci-dessous), et supposons qu'un jeton circulent indéfiniment dans ce système. Supposons que les processus n’ont aucune idée du nombre de jetons circulant dans le réseau. Nous voulons que les processus coopèrent les uns avec les autres pour compter le nombre exact de ces jetons (sans jamais arrêter le système). La tâche doit être lancée par un processus initiateur (disons le processus 0) qui enverra des messages d'interrogation aux autres processus pour enregistrer le nombre de jetons qu'ils ont vus. Nous désignons par le nombre de jetons enregistrés par le processus . Montrer que selon le moment où les processus individuels comptent les jetons, plusieurs situations sont possibles.

0

2

1

**Possibilité 1 :** Le processus 0 enregistre lorsqu'il reçoit le jeton. Lorsque le processus 1 enregistre

n1, on suppose que le jeton est dans le canal (1, 2), donc . Lorsque le processus 2 enregistre , le jeton est dans le canal (2, 0), donc . Par conséquent, .

**Possibilité 2 :** Le processus 0 enregistre quand il reçoit le jeton. Le processus 1 enregistre lorsque

le jeton a atteint le processus 1, donc . Enfin, le processus 2 enregistre lorsque le jeton

a atteint le processus 2, donc . Ainsi, . Puisque les jetons sont indiscernables, aucun processus ne sait que le même jeton a été compté trois fois !

**Possibilité 3 :** Le processus 0 enregistre puisque le jeton est dans le canal (0, 1) au moment de l'enregistrement. Lorsque les processus 1 et 2 veulent enregistrer le comptage, les jetons les ont déjà quittés.

donc et . Ainsi, **.**

**Exercice 2**

Supposons que les horloges physiques de tous les processus soient synchronisées et que les délais de propagation des canaux soient connus. Proposez un algorithme alternatif pour le calcul d'un instantané distribué, basé sur les horloges physiques et les délais des canaux.

Rappelons que nous avons deux hypothèses :

1. Horloges synchronisées
2. Délais de propagation sur les canaux connus

* Donc, un processus choisi (p0, par exemple) fixe une date ( h ) de prise de snapshot et la diffuse à tous les autres processus.
* Lorsque l’horloge d’un processus donné indique h, celui-ci enregistre son état et indique pour chaque canal sortant les messages non encore reçus par les processus correspondants.
* Tant que l’horloge d’un processus donné n’est pas encore arrivée à h, celui-ci enregistre dans une file les messages envoyés sur chaque canal sortant qui n’arriveront pas avant h à leurs destinataires.

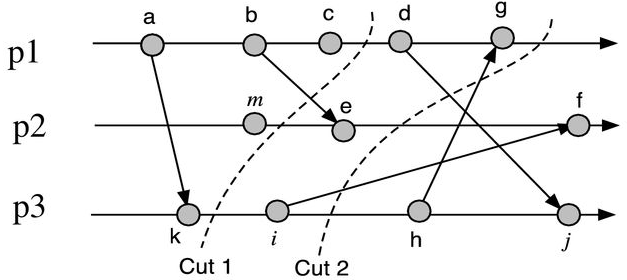
La précaution à prendre avec cet algorithme sont :

* choisir h de telle sorte que sa diffusion soit reçue par tous les processus avant h.
* Chaque processus qui envoie un message sur un canal, de sortie, enregistre ce message avec la date de réception prévue.
* Chaque message est retiré de la file à la date de sa réception prévue. On peut déterminer les messages à retirer de la file à chaque envoie d’un nouveau message, par exemple. A l’heure h, cette opération est aussi effectuée.

**Exercice 3**

Dans la figure ci-dessous, montrez toutes les coupes cohérentes qui

(a) incluent l'événement d

(b) excluent l'événement d mais incluent l'événement g.

Rappel :

Une coupure est un ensemble d'événements ; elle contient au moins un événement par processus. Dessinez une ligne de temps pour chaque processus dans un système distribué, et représentez les événements par des points sur la ligne de temps, comme indiqué sur la figure. Ici, {c, d, f, g, h} est une coupe.

Une coupe est dite cohérente si, pour chaque événement qu'elle contient, elle inclut également tous les événements précédant cet événement dans l'ordre causal.

Soit a et b deux événements dans un système distribué ; alors,

Ainsi, pour un message m, si l'état suivant receive(m) appartient à une coupe cohérente, alors l'état suivant send(m) doit aussi appartenir à cette coupe.

2. Impossible car donc d doit être inclus dans toute coupure cohérente qui contient g.

**Exercice 4**

Considérons l'algorithme d'instantanés distribués de Chandy-Lamport s'exécutant sur un réseau dont la topologie est, où est l'ensemble des nœuds représentant les processus et est l'ensemble des arêtes dirigées (représentant les canaux) reliant les paires de processus. Soit . Prouvez que dans l'instantané, les états de canaux seront toujours vides.

Rappel : Algorithme Chandy-Lamport (snapshot)

Démarrage :

* un certain processeur P0 s'envoie un jeton "take a snapshot" (prendre un instantané)

Progression :

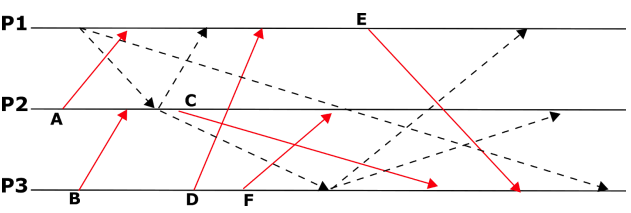
* la première fois que Pi reçoit un jeton "take a snapshot" (de Pj)
* Pi enregistre son état local
* Pi envoie "take a snapshot" sur tous les canaux sortants
* Pi enregistre que le canal de Pj est "vide".
* Pi commence à enregistrer les messages de tous ses autres canaux.
* à partir de ce moment, lorsque Pi reçoit "take a snapshot" de Pk
* Pi arrête d'enregistrer le canal de Pk

Terminaison :

- lorsque Pi a reçu "take a snapshot" sur tous les canaux entrants, il envoie son état enregistré à P0 et s'arrête.

Chaque processus Pi, lorsqu’il reçoit pour la première fois un marqueur ("take a snapshot") de la part d’un processus Pj, il met l’état du canal (i, j) à vide. Puisque tous les processus P1, P2, P2, … , Pn-1 reçoivent, chacun, un tel marqueur une première fois, donc ils vont, chacun, initialiser un canal entrant à vide. Donc au moins n-1 canaux vont être vides.

**Exercice 5**

En utilisant l’algorithme de Chandy-Lamport (pour la prise d’instantané) montrer quand chaque processus enregistre-t-il son état local et lister les états des canaux pour chaque processus capturé dans l’instantané. Les lignes en pointillés sont des messages marqueurs. Les lignes rouges sont des messages ordinaires (A à F).