1. INTRODUCTION.

Dans le domaine de l'algorithmique distribuée de nombreux algorithmes, dont le but est de réaliser un contrôle du système réparti dans lequel ils apparaissent, sont basés sur l'existence d'un message spécial appelé **jeton**. Lorsque le jeton est unique, sa possesion par un processus assure à ce dernier une exclusivité qui peut être mise à profit pour réaliser une discipline de contrôle particulière; citons à titre d'exemples la réalisation d'une exclusion mutuelle [LEL77, RIC83], le droit de modifier une copie de fichier dupliqué [LEL78] et la réalisation d'une élection [CHA79, DOL82].

Les algorithmes basés sur l'existence d'un jeton ont souvent l'advantage d'être simples. Cette simplicité est réelle lorsque les lignes de communication sont fiables, c'est-à-dire lorsqu'il n'y a ni altérations ni pertes de messages. Il n'en est plus de même lorsque les lignes ne sont pas fiables car la perte du jeton est alors fatale à l'algorithme distribué qui ne peut plus progresser. Deux problèmes se posent alors au concepteur de l'algorithme : détecter la perte du jeton et, si tel est le cas, le regénérer.

La perte du jeton peut a priori être détectée par un ou plusieurs processus ; dans tous les cas il ne faut en regénérer qu'un exemplaire unique. Plusieurs solutions ont été proposées pour résoudre ces problèmes ; la plupart s'appuient sur une topologie de communication entre processus particulière et sur l'utilisation des mêmes méthodes. Considérons d'abord l'aspect structurel des lignes. Il n'y a pas de processus privilégié vis a vis du jeton ; en conséquence la topologie des communications est non-hierarchique et tout processus doit pouvoir communiquer (directement ou non) avec tout autre processus ; les structures considérées, dans les algorithmes basés sur les jetons, sont donc l'anneau et le maillage complet. Pour la détection de la perte, la méthode couramment utilisée est l'emploi d'horloges de garde [LEL77]. Lorsqu' un processus transmet le jeton à son successeur sur l'anneau par exemple, il arme une horloge de garde. Le calibrage du délai correspond au temps au bout duquel ce processus devrait avoir à nouveau reçu le jeton. Lorsque le délai est écoulé le processus entame une négociation avec les autres processus placés sur l'anneau afin de determiner si le jeton est effectivement perdu (cette phase peut être recommencée plusieurs fois si les messages relatifs à la négociation se perdent). Il est crucial de ne regénérer le jeton que s'il est perdu et de ne le regénérer alors qu'en un seul exemplaire ; pour

assurer cette unicité les identités des processus (qui doivent être toutes distinctes) sont utilisées.

Les algorithmes de détection de la perte de jeton et de regénération sont ainsi généralement complexes ; utilisant l'identité des processus ils présentent un caractère dissymétrique (les processus n'ont le même texte qu'à l'identité prés et en conséquence ont des comportements différents) et la panne des processus peut à nouveau leur être fatale.

Ce rapport présente des solutions au problème de la perte d'un jeton (détection et regénération) qui ne souffrent pas des inconvénients précédents, puisqu'ils n'utilisent ni horloge de garde ni identité des processus. La Section 2 présente un algorithme dû à Misra qui utilise deux jetons. Dans la Section 3 nous présentons un nouvel algorithme distribué et symétrique à deux jetons fondé sur un principe différent que nous généralisons à k jetons dans la Section 4 pour améliorer ses performances. Le principe sur lequel est fondé l'algorithme consiste à utiliser des compteurs monotones croissants.

2. L'ALGORITHME DE MISRA.

Hypothèses.

Misra a proposé dans [MIS83] un algorithme de détection de perte de jeton et de regénération qui n'utilise ni horloge de garde, ni identité des processus. Avant de l'énoncer rappelons les hypothèses contextuelles sur lesquelles il s'appuie (ces hypothèses sont valables pour tous les algorithmes présentés dans ce rapport). Les processus sont situés sur un anneau uni-directionnel (du point de vue du transfert du jeton et donc du privilège associé). Les lignes de communication sur cet anneau peuvent perdre les messages mais ne les altèrent pas. De plus les messages ne peuvent pas se doubler sur une ligne de communication (mais le peuvent à l'intérieur d'un processus).

Principe.

La solution proposée par Misra repose sur un principe simple : elle consiste à utiliser deux jetons dont chacun sert à détecter la perte éventuelle de l'autre. Le procédé adopté est le suivant : un jeton arrivé à P_i peut garantir que l'autre jeton est perdu (et alors le regénérer) si depuis le passage précédent de ce jeton à P_i, ni lui ni le processus P_i n'ont rencontré l'autre jeton. Les deux jetons ont des comportements symétriques, chacun permettant de détecter la perte de l'autre (du point de vue du privilège associé au jeton, un seul d'entre eux est pertinent). La perte d'un jeton est détectée par l'autre en un tour sur l'anneau. Cet algorithme ne fonctionne donc que si, un jeton étant perdu, l'autre effectue un tour complet sans se perdre. Nous proposons dans la Section 4 un algorithme utilisant un nombre quelconque de jetons ; dans ce cas, l'algorithme fonctionne tant qu'il reste au moins un jeton sur l'anneau. Dans chaque contexte particulier on peut a priori, par un choix conséquent du nombre de jetons, rendre la probabilité de défaillance de l'algorithme aussi faible que l'on veut (Rappelons que les algorithmes "classiques" ne sont pas exempts de ce problème, les messages de négociation entre processus pouvant être indéfiniment perdus).

L'algorithme.

Soient j_0 et j_1 les deux jetons. Leurs passages dans chaque processus et leurs rencontres devant être mémorisées, on les value par un entier qui est transporté par les jetons : vj_0 et vj_1 respectivement ; la valeur absolue de chacun d'eux compte le nombre de fois où les jetons se sont rencontrés, ces

deux nombres étant liés par l'invariant :

$$vi_0 + vi_1 = 0 \ .$$

Initialement les jetons sont dans un processus quelconque de l'anneau et l'on a $vi_0 = 1$ et $vi_1 = -1$. Chaque processus P_i est doté d'une variable entière m qui mémorise la valeur associée au dernier jeton qu'a vu passer le processus :

var m : **entier initialisé à** 0.

Les processus (sites) étant les seules entités où peuvent être effectués des calculs, l'algorithme est décrit par les évenements (et les traitements associés) dont les processus sont les sièges. Le comportement de chaque processus P_i est le suivant :

lors de

la réception de (i_0, vi_0)
faire
si $m = vi_0$ **alors**
début
(* i_1 est perdu : le regénérer *)
$vi_0 := vi_0 + 1$;
$vi_1 := -vi_0$
fin
sinon $m := vi_0$
fsi
fait

la réception de (i_1, vi_1)
faire
(* traitement analogue au précédent en intervertissant les rôles de i_0 et i_1 *)
fait

la rencontre de i_0 **et** i_1
faire
$vi_0 := vi_0 + 1$;
$vi_1 := vi_1 - 1$
fait

Commentaires.

L'algorithme consiste à maintenir toujours vraie la relation $vi_0 + vi_1 = 0$. Pour cela lorsque les deux jetons se rencontrent (rappelons qu'ils ne peuvent se rencontrer que dans un processus) on modifie les valeurs qui leur sont associées en conséquence. Lorsque P_i reçoit l'un des jetons, il teste si sa variable m a la valeur du jeton reçu, soit vi_0. Si $m \neq vi_0$, le jeton j_1 soit est nécessairement passé par P_i, soit a rencontré j_0 : il n'est donc pas perdu : P_i mémorise alors le passage de j_0 par l'instruction $m := vi_0$. Dans le cas où à la réception de j_0 le site P_i trouve m égal à vi_0, d'une part le jeton j_1 n'est pas passé par P_i (sinon m aurait été modifié en conséquence) et d'autre part, les deux jetons ne se sont pas rencontrés sur l'anneau (sinon le compteur vi_0 aurait une valeur supérieure à son ancienne valeur) : en conséquence le jeton j_1 est perdu. Le processus P_i le regénére en affectant aux deux compteurs vi_0 et vi_1 les valeurs adéquates.

Borner le domaine des compteurs.

La taille des compteurs vi_0 et vi_1 associés aux jetons n'est pas bornée a priori, ce qui peut constituer un inconvénient pour l'implémentation de cet algorithme. Nous allons voir que les propriétés de l'algorithme permettent de borner ces compteurs.

L'algorithme n'utilise en effet que des tests de comparaisons sur les entiers, du type égalité/inégalité et non du type inférieur/supérieur (ce qui de plus en autorise une mise en oeuvre matérielle simple). Lorsque les compteurs sont mis à jour (lors d'une rencontre ou d'une regénération) ils sont incrementés (en valeur absolue) et prennent alors des valeurs différentes de celles prises par les variables m_i des processus P_1 à P_n. Il est donc possible d'incrémenter les compteurs vi_0 et vi_1 modulo $n+1$: vi_0 par exemple, ne pourra pas alors prendre la même valeur que l'une des variables m_i : pour que cela se produise il faudrait que vi_0 aitt été incrementé $n+1$ fois depuis sa dernière visite à P_i ce qui est impossible puisqu'il n'y a que n processus et que les jetons ne se rencontrent que dans les processus et s'y rencontrent au maximum une fois.

Compteurs à valeurs booléennes.

Dans l'algorithme de Misra que nous venons de présenter l'identité des processus n'intervient pas : ceux-ci peuvent être placés de manière quelconque sur l'anneau, ce qui constitue une propriété intéressante, aucun processus ne

jouant un rôle privilégié. Toutefois si l'identité des processus n'intervient pas il n'en est pas de même de leur nombre *n* lorsque l'on borne le domaine de définition des compteurs : ce dernier est fonction du nombre de processus placés sur l'anneau. Est-il possible d'éliminer cette contrainte ?

Considérons l'algorithme précédent et prenons comme compteurs des booléens. Chacun des compteurs est respectivement initialisé à *vrai* et *faux* (les variables *m* étant initialisées à la valeur *nil*). La relation à conserver invariante entre les compteurs est donc vj_0 = **non** vj_1 : lors d'une rencontre les jetons échangent leurs valeurs de façon à mémoriser cette rencontre. Pour voir si l'algorithme est correct, plaçons-nous du point de vue d'un processus P_i quelconque sur l'anneau : si les jetons ne se perdent pas, P_i doit voir passer la séquence de valeurs *(vrai ; faux)** indépendamment des jetons qui véhiculent ces valeurs. Pour qu'il en soit ainsi il est nécessaire que lorsque les jetons se rencontrent dans un processus, le dernier jeton arrivé double l'autre jeton. En effet ceci assure que les valeurs *vrai / faux* véhiculées par les jetons seront vues par chaque processus selon la séquence *(vrai ; faux)**. Si par contre lors d'une rencontre les jetons peuvent ou non se doubler l'algorithme devient incorrect, des regénérations de jetons non perdus pouvant se produire.

Conclusion

En conclusion rappelons les principales propriétés de l'algorithme de Misra : dans une structure de processus connectés en anneau, la détection de la perte d'un jeton est effectuée sans utiliser ni horloge de garde ni identité des processus, seul le nombre de processus intervient lorsque l'on borne le domaine de définition des compteurs manipulés par l'algorithme. Il est possible d'éliminer cette dépendence et de ne considérer que des valeurs booléennes pour les jetons mais lors de leurs rencontres ils doivent alors obligatoirement se dépasser ; ceci peut par exemple correspondre à une gestion LIFOP (Last In First Out with Preemption, i.e. Dernier Arrivé Premier Sorti avec Préemption) des jetons dans chaque site.

3. UN ALGORITHME A 2 JETONS BASE SUR LE COMPTAGE.

Nous proposons un algorithme à 2 jetons fondé sur un principe différent de celui de Misra que nous généraliserons à un nombre quelconque de jetons dans la Section 4. Comme précédemment les jetons tournent dans le même sens sur l'anneau des processus : .... P_i, P_{i+1}, ... (les opérations sur les indices des processus sont réalisées modulo n).

Principe.

L'algorithme est basé sur l'idée suivante : lorsqu'un jeton h passe dans un site P_i il mémorise son passage dans une variable locale de P_i en lui affectant la valeur d'un compteur qu'il transporte, après avoir incrémenté ce compteur ; celui-ci précise à tout instant le nombre de sites visités par ce jeton. La trace laissée par un jeton, lors de son passage en P_i, permettra à l'autre jeton de détecter sa perte éventuelle lorsqu'il arrivera en P_{i+1}.

La détection de la perte du jeton h entre les sites P_i et P_{i+1} par l'autre jeton j se fait de la manière suivante. Outre son compteur le jeton j véhicule la valeur de la trace qu'a laissée le jeton h lors de son passage en P_i. Lorsque j arrive en P_{i+1}, il doit trouver la trace qu'y a laissé le jeton h égale à la trace que h a laissé en P_i (et que j transporte) augmentée de un (puisque h a visité un site de plus) ; si tel n'est pas le cas, le jeton h s'est perdu entre P_i et P_{i+1}, la détection de la perte ayant donc lieu en P_{i+1}.

Pour que le principe soit correct il reste à examiner le comportement que doivent présenter les deux jetons lorsqu'ils se rencontrent car ils peuvent alors éventuellement se doubler. Si tel est le cas, soit j double le jeton h, il va trouver une rupture de séquence dans les traces laissées par h dans les processus ; celle-ci ne doit pas l'amener à conclure à la perte de h. Pour cela, chacun des jetons est doté d'un drapeau s_j qu'il met à vrai lorsqu'il rencontre l'autre jeton. En conséquence, lorsque j arrive dans un processus il se comportera comme nous l'avons vu si son drapeau est à *faux* ; si par contre s_j est à *vrai* le jeton j met à jour l'image de la trace de h qu'il véhicule et repositionne son drapeau à *faux*. L'introduction des drapeaux est donc dûe au fait que les jetons ont des vitesses de rotation indépendantes et a priori différentes dans l'anneau des processus et lorqu'ils se rencontrent dans l'un des sites ils peuvent s'y doubler.

L'algorithme.

Chaque jeton se présente comme un triplet (j, v_j, s_j) dans lequel :

* j désigne son identité ($j = 0,1$).
* v_j est un tableau à 2 entrées :
  - $v_j[j]$ compte le nombre de processus visités par j.
  - $v_j[h]$ mémorise la dernière valeur de la trace laissée par le jeton h ($h = j + 1$ **mod** 2) dans le dernier processus visité par j.
* s_j est le booléen mis à *vrai* lors d'une rencontre.

Chaque processus P_i est doté d'un tableau d'entiers m à deux entrées correspondant aux deux jetons j et h : ce tableau permet à chacun des jetons de laisser la trace de son passage dans le site.

Au début les jetons sont placés de façon quelconque dans l'anneau des processus : les champs v_j et s_j des jetons et les variables m_j des processus sont initialisés en conséquence (voir la Section 4 où les initialisations sont décrites dans le cas général d'un nombre quelconque de jetons).

Le comportement d'un processus P_i quelconque est décrit par l'algorithme suivant :

(page suivante)

```
lors de la réception de ( j, v_j, s_j )
    faire
        v_j[j] := v_j[j] + 1;
        m[j] := v_j[j];
        h := j + 1 mod 2;
        si h présent alors
            début «rencontre des deux jetons»
                v_j[h] := m[h]; s_j := vrai;
                v_h[j] := m[j]; s_h := vrai
            fin
(α)     sinonsi m[h] ≠ v_j[h] + 1 et non s_j alors
            début «regénération du jeton h»
                v_h := v_j;
                «simulation arrivée de h»
                v_h[h] := v_h[h] + 1;
                m[h] := v_h[h];
                «simulation de la rencontre»
                v_j[h] := m[h]; s_j := vrai;
                v_h[j] := m[j]; s_h := vrai
            fin
            sinon
                début
(β)                 v_j[h] := m[h]; s_j := faux
                fin
            fsi
        fsi
    fait
```

Commentaires.

La détection de la perte d'un jeton (ici noté h) a lieu à la ligne α : le drapeau s_j est à *faux* et il y a rupture de séquence dans la trace de h. Le jeton perdu est regénéré et les contextes des jetons et du processus dans lequel la regénération a lieu sont mis à jour (simulation de l'arrivée du jeton regénéré et de la rencontre entre les deux jetons). S'il n'y a pas rupture de séquence où s'il y a eu possibilité de dépassement des jetons il n'y a pas détection de (fausse) perte mais seulement des mises à jour (ligne β). Les mises à jour redondantes sont laissées par souci de clarté dans l'exposé de

l'algorithme.

Borner le domaine des compteurs.

De la même façon que dans le cas de l'algorithme de Misra, on peut faire évoluer les compteurs associés aux deux jetons (i.e. $v_j[j]$ et $v_h[h]$) modulo $n+1$ en évitant ainsi les problèmes d'implémentation associés à l'absence de bornes.

Remarque.

Si un jeton s'est perdu entre les processus P_{i-1} et P_i et si les deux jetons n'étaient pas simultanément présents en P_{i-1} lorsque l'un d'eux est parti et s'est perdu, alors cette perte et la regénération associée auront lieu en P_i, donc "au plus tôt", ce qui n'est pas le cas de l'algorithme de Misra.

Dans le cas où j et h se trouvent simultanément dans un site et le premier à le quitter se perd, il y a un moyen simple d'éviter le tour complet sur l'anneau de l'autre jeton pour détecter la perte : si h est le premier jeton à partir du site, alors lors de son départ le processus met le drapeau de j à *faux* et de cette façon ce dernier en détectant la rupture de séquence saura que h vient de se perdre (voir Section 4).