RAIRO. INFORMATIQUE THÉORIQUE

MAURICE TCHUENTE

# Sur l'auto-stabilisation dans un réseau d'ordinateurs

*RAIRO. Informatique théorique*, tome 15, n° 1 (1981), p. 47-66

<http://www.numdam.org/item?id=ITA_1981__15_1_47_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 15, n° 1, 1981, p. 47 à 66)

# SUR L'AUTO-STABILISATION DANS UN RÉSEAU D'ORDINATEURS (*)

par Maurice TCHUENTE (1)

Communiqué par J.-F. PERROT

Résumé. — *On étudie le nombre minimal d'états nécessaires à l'auto-stabilisation sur une boucle et sur une chaîne d'automates finis, ce qui fait apparaître le caractère optimal de deux algorithmes de E. W. Dijkstra. Ensuite, on présente des méthodes récurrentes permettant de construire sur un réseau connexe donné, une classe d'algorithmes d'exclusion mutuelle auto-stabilisants, définis à partir d'algorithmes associés à des sous-réseaux qui le recouvrent; dans le cas particulier d'une chaîne, on exhibe un algorithme récurrent qui se stabilise deux fois plus vite que celui proposé par E. W. Dijkstra.*

Abstract. — *Minimum-state solutions to self-stabilization in a ring and in a chain of finite state automata are studied; the results show that two of E. W. Dijkstra's algorithms are optimal. Further, we present recurrent algorithms for the construction of self-stabilizing networks of finite state automata, which are based on algorithms defined on sub-networks; for the particular case of a chain, we exhibit a recurrent algorithm whose stabilization time is two times less than that of E. W. Dijkstra.*

## INTRODUCTION

Considérons un ensemble de $N+1$ ordinateurs $M_i$ $(i=0, \ldots, N)$ ayant entre eux un réseau limité de connexions représenté par un graphe $G$ d'ordre $N+1$; deux d'entre eux $M_i$, $M_j$ sont dits voisins si les sommets $i$, $j$ sont adjacents dans $G$.

On suppose que ces machines doivent partager une ressource commune non utilisable simultanément par deux d'entre elles; une machine autorisée à accéder à cette ressource est dite privilégiée. Il est alors nécessaire de définir un algorithme qui, en cas de demandes simultanées, accorde le privilège en exclusion mutuelle, c'est-à-dire à tour de rôle. Chaque machine $M_i$ doit alors, en plus des variables affectées à son programme particulier, disposer d'une variable de coopération $x_i$.

(*) Reçu mars 1979, révisé décembre 1979.
(1) Laboratoire I.M.A.G., Grenoble.

Dans le cas d'un système centralisé, toutes les variables de coopération sont rangées dans une mémoire centrale; $x_i$ est accessible en lecture pour toutes les machines du réseau mais ne peut être modifié que par $M_i$ et, de plus, on peut supposer l'existence d'une variable globale accessible en lecture-écriture à toutes les machines. Par contre, dans le cas des systèmes répartis qui nous intéressent, il n'y a pas de mémoire centrale et aucune variable n'est accessible en lecture-écriture à plusieurs machines; de plus, la variable $x_i$, qui est gérée par $M_i$, ne peut être lue que par les machines voisines.

Si on a par exemple un réseau en boucle, et si deux machines $M_i$, $M_j$ ont des voisinages disjoints, on peut, pour tout algorithme d'exclusion mutuelle, trouver des configurations pour lesquelles elles sont simultanément privilégiées. E. W. Dijkstra, dans son article intitulé « Self-stabilizing systems in spite of distributed control » [1] a donc introduit, pour les algorithmes d'exclusion mutuelle dans un système réparti, la propriété importante d'auto-stabilisation; cela signifie que, pour toute configuration de départ et pour toute évolution au cours de laquelle chaque machine est privilégiée une infinité de fois, le système doit au bout d'un temps fini, arriver à un régime d'exclusion mutuelle.

Ici, nous nous intéressons d'abord à la complexité de la propriété d'auto-stabilisation; pour cela nous étudions, dans les cas de réseaux en boucle ou en chaîne, le nombre minimal d'états possibles dont chaque machine doit disposer pour que l'auto-stabilisation soit réalisable.

Nous introduisons ensuite des méthodes récurrentes permettant de construire sur un réseau connexe donné, une classe d'algorithmes d'exclusion mutuelle auto-stabilisants définis à partir d'algorithmes associés à des sous-réseaux qui le recouvrent. L'intérêt de cette approche réside notamment dans le fait qu'elle conduit directement au théorème d'existence de solution pour un réseau connexe quelconque, à partir de la solution triviale associée à un couple de machines. Comme exemple d'application, nous exhibons, dans le cas d'une chaîne, un algorithme optimal du point de vue du nombre de variables d'état; cet algorithme se stabilise deux fois plus vite que celui proposé par E. W. Dijkstra, phénomène qui se justifie intuitivement par le fait que, contrairement à celui présenté dans [1], il garantit toujours l'exclusion mutuelle entre deux machines voisines.

## 0. NOTATIONS ET DÉFINITIONS

Soit $G=(S, \Gamma)$ un graphe non orienté tel que :

- $S=\{0, 1, \ldots, N\}$ est l'ensemble des sommets;
- pour $i \in S$, $\Gamma(i)=\{i_1, i_2, \ldots, i_r\} \subset S$ désigne les sommets adjacents à $i$.

En chaque sommet $i \in S$ est placé un automate fini $M_i$ tel que :

– $Q_i$ est l'ensemble des états ($Q_i$ fini non vide);

– si $\Gamma(i) = \{i_1, i_2, \ldots, i_r\}$, $i_1 < i_2 < \ldots < i_r$, alors l'ensemble des entrées est constitué par les états des machines voisines :

$$Q_{i_1} \times Q_{i_2} \times \ldots \times Q_{i_r};$$

– sa fonction de transition est une application :

$$\delta_i : \quad Q_i \times Q_{i_1} \times Q_{i_2} \times \ldots \times Q_{i_r} \to Q_i.$$

Un état (ou une configuration) du système, $x \in \prod_{i=0}^{N} Q_i$ se note $x = x_0\, x_1 \ldots x_N$ où $x_i$ désigne l'état de $M_i$.

Une machine $M_i$ est dite privilégiée (ou activable) dans la configuration $x = x_0\, x_1 \ldots x_N$ si elle peut changer d'état :

$$\delta_i(x_i, x_{i_1}, x_{i_2}, \ldots, x_{i_r}) \neq x_i \qquad [\Gamma(i) = \{i_1, i_2, \ldots, i_r\}].$$

On dit qu'une configuration est légitime si elle admet exactement une machine privilégiée; une configuration n'admettant aucune machine privilégiée est dite de blocage.

Si $x$ n'est pas une configuration de blocage, et si $\{M_i, i \in I\}$ est l'ensemble des machines privilégiées dans $x$, on appelle $\Delta$-transition partielle, ou activation, le passage de $x$ à un état $y$ tel que :

$$y = y_0\, y_1 \ldots y_N; \qquad y_i = \begin{cases} x_i & \text{si } i \notin J, \\ \delta_i(x_i, x_{i_1}, \ldots, x_{i_r}); & \Gamma(i) = \{i_1, \ldots, i_r\},\ i \in J, \end{cases}$$

avec $J$ partie non vide de $I$.

On dit alors que $\{M_j, j \in J\}$ est l'ensemble des machines activées lors du passage de la configuration $x$ à la configuration $y$. Lorsque $I$ a plus d'un élément, il y a plusieurs activations possibles à partir de $x$, mais si $x$ est une configuration légitime, l'unique $\Delta$-transition définissable à partir de $x$ est celle qui active l'unique machine privilégiée dans $x$.

Une séquence d'activations est dite régulière si chaque machine $y$ est activée une infinité de fois.

Le système est dit en exclusion mutuelle si :

– il est dans une configuration légitime;

– partant de $x$, on peut définir une et une seule séquence régulière d'activations et de plus dans cette unique séquence, chaque machine est activée une infinité de fois.

Un algorithme d'exclusion mutuelle est dit auto-stabilisant si :

– à partir de toute configuration initiale, il existe au moins une séquence régulière d'activations;

– pour toute séquence régulière d'activations associée à une configuration initiale $x$, on aboutit en un nombre fini de pas à un régime d'exclusion mutuelle.

Évidemment, un tel algorithme n'admet pas de configuration de blocage. Le temps de stabilisation d'un tel algorithme est le nombre maximal de pas qu'il faut accomplir avant d'atteindre le régime d'exclusion mutuelle, ce maximum étant pris par rapport à toutes les configurations de départ et à toutes les séquences régulières d'activations.

Il est clair que, dans la pratique, on recherchera des algorithmes pour lesquels le nombre de variables d'état, le temps de stabilisation et le nombre maximal de changements d'état avant le régime d'exclusion mutuelle, sont aussi petits que possible.

## 1. EXISTENCE DE SOLUTION SUR TOUT RÉSEAU CONNEXE

Nous allons montrer comment l'algorithme de Dijkstra défini sur une boucle, permet de déduire l'existence d'une solution pour tout réseau connexe.

Soit donc la boucle :

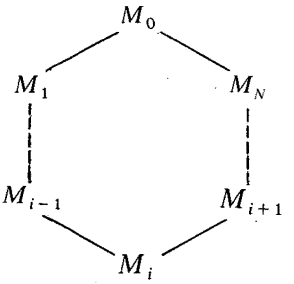



**Figure 1**

Chaque machine $M_i$ est reliée à $M_{i-1}$ et $M_{i+1}$, les indices étant calculés modulo $N$.

*Description de l'algorithme*

Les fonctions de transition sont écrites comme des instructions du type algol $W$. Toutes les machines ont pour ensemble d'états :

$$\{0, 1, 2, \ldots, K\}, \quad \text{où } K \text{ est un entier,} \qquad K > N;$$

machine $M_0$ :

$$\text{si} \quad x[0]=x[N] \text{ alors } x[0]:=x[0]+1 \text{ modulo } K;$$

machine $M_i$; $1 \leqq i \leqq N$ :

$$x[i]:=x[i-1].$$

PROPRIÉTÉS : 1. Il n'y a pas de configuration de blocage.

2. A partir d'une configuration quelconque $x$, l'application d'une séquence régulière d'activations permet d'obtenir après au plus $N$ changements d'état de $M_0$, une configuration de la forme $a^{N+1}$.

3. Le nombre maximal de changements d'état de $M_i$ avant qu'on n'atteigne une configuration de la forme $a^{N+1}$ vérifie :

$$\varphi(0)=N; \qquad \varphi(i)=\varphi(i-1)+1, \quad i=1, \ldots, N.$$

*Démonstration :* 1. Facile à vérifier.

2. Soit $x$ une configuration de la forme :

$$x=\alpha_0^{n_0}\alpha_1^{n_1}\ldots\alpha_p^{n_p}, \quad n_0>0, \qquad n_0+n_1+\ldots+n_p=N+1.$$

Les $\alpha_i^{n_i}$ tels que $n_i>0$ sont appelés composantes de $x$. Après $N$ changements d'état de $M_0$, on a nécessairement une configuration de la forme :

$$(\alpha_0+N)^{m_0}(\alpha_0+N-1)^{m_1}\ldots\alpha_0^{m_N}, \quad m_0>0; \qquad m_i \geqq 0, \quad 1 \leqq i \leqq N.$$

Comme $K>N$, on a :

$$\alpha_0+i \neq \alpha_0+N, \qquad i=0, \ldots, N-1$$

et $M_0$ ne peut à nouveau changer d'état que dans la configuration $(\alpha_0+N)^{N+1}$.

3. $\varphi(0)=N$ d'après ce que nous avons démontré plus haut. Pour $i \geqq 1$, $\varphi(i)$ est le nombre maximal d'états différents que $M_{i-1}$ peut avoir pendant la période considérée, et vaut donc $\varphi(i-1)+1$.

PROPOSITION : *L'algorithme est auto-stabilisant. De plus, avant qu'on n'atteigne le régime d'exclusion mutuelle, on peut avoir au plus* $(3/2)N(N+1)-1$ *changements d'état et le temps de stabilisation est* $(N(N+3)/2)-2$.

*Démonstration :* Les propriétés démontrées plus haut entraînent que, pour toute séquence régulière d'activations on obtient au bout d'un nombre fini de pas, une configuration ayant au plus deux composantes, et on vérifie aisément qu'on est alors dans le régime d'exclusion mutuelle.

Avant la stabilisation, les configurations ont au moins trois composantes, et si de plus, toute activation conduit à une configuration légitime, on a nécessairement :

$$x = a^{N-1}bc, \qquad a, b, c \text{ distincts.}$$

En utilisant les propriétés démontrées précédemment, on voit que le nombre total de changements d'état avant la stabilisation est inférieur ou égal à :

$$\varphi(0)+\varphi(1)+\ldots+\varphi(N-1)+\varphi(N)-1=\frac{3}{2}N(N+1)-1.$$

Pour atteindre cette borne, on part de :

$$x=a(a+N-1)(a+N-2)\ldots(a+1)a.$$

Pendant $N$ étapes consécutives on active simultanément toutes les machines, et on continue avec la séquence :

$$N, N-1, N, \ldots, i, (i+1), \ldots, N, \ldots,$$
$$2, 3, \ldots, N-1, 1, 2, \ldots, N-2, \{N-1, N\}.$$

On vérifie que ceci correspond aussi au nombre maximal d'étapes avant la stabilisation, *cf.* [2].

COROLLAIRE : *Il existe, pour tout réseau connexe, un algorithme d'exclusion mutuelle auto-stabilisant.*

*Démonstration :* Soit $s_0 s_1 s_2 \ldots s_m s_0$, $m \geqq N$ un cycle non nécessairement élémentaire, passant par tous les sommets du graphe $G$ sous-jacent au réseau. A partir d'un algorithme défini sur cette boucle virtuelle et utilisant les variables $y_j$, $j=0, \ldots, m$, on définit sur le réseau, un algorithme en associant à chaque $M_i$ un vecteur état :

$$x_i=(y_{j_1}, y_{j_2}, \ldots, y_{j_s}),$$

où $\{j_1, \ldots, j_s\}$ est l'ensemble des occurences du sommet $i$ dans le cycle.

Il est clair que l'auto-stabilisation qui est acquise pour les $y_j$ le sera *a fortiori* pour les variables $x_i$.

COMMENTAIRE : Comme le montre la figure ci-dessous, la boucle virtuelle définie au corollaire précédent peut avoir beaucoup plus de sommets que le graphe initial, ce qui se traduit notamment par un temps de stabilisation très long. Les algorithmes récurrents que nous allons présenter au paragraphe 4 permettent de mieux tenir compte des caractéristiques du réseau.

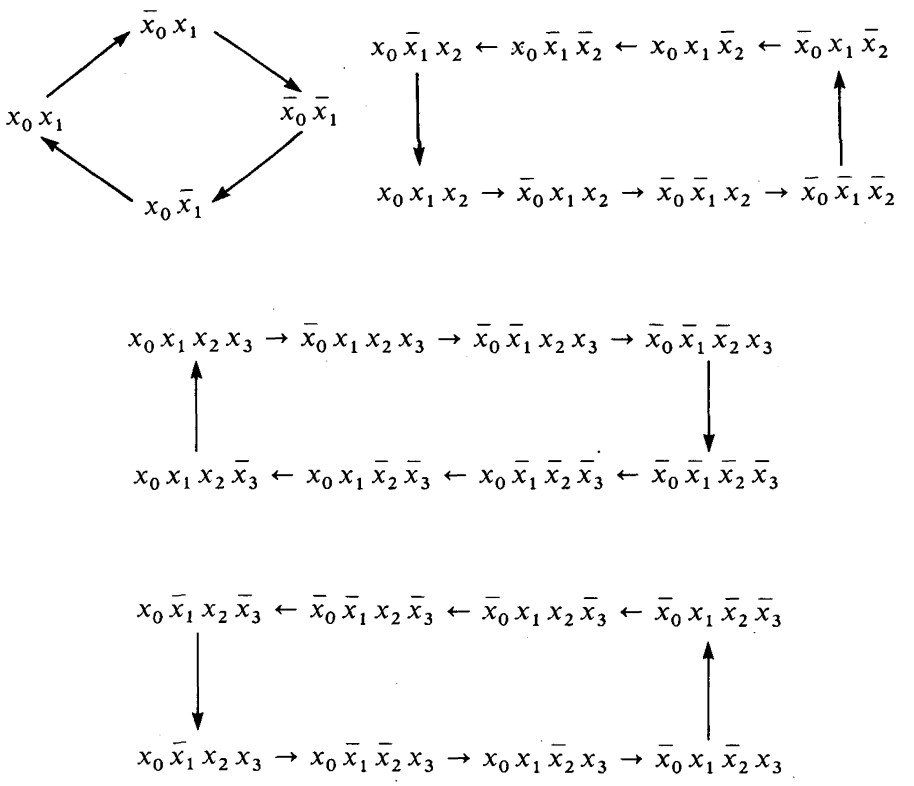
**Figure 2**

## 2. CAS D'UNE BOUCLE

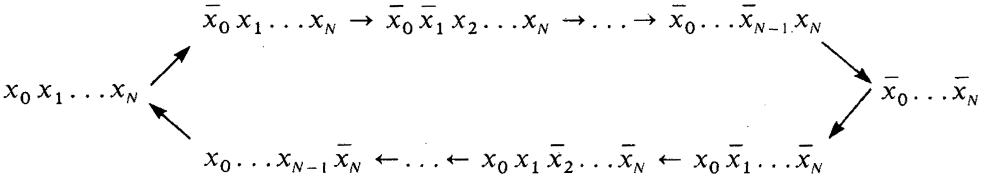
Dans ce paragraphe, on étudie le nombre minimal d'états nécessaires à l'auto-stabilisation sur une boucle.

PROPOSITION 2.1 : *Soit une boucle de $N+1$ machines dont chacune a deux états possibles. On peut y définir un algorithme d'exclusion mutuelle auto-stabilisant si et seulement si $N \leqq 3$.*

*Démonstration :* Condition suffisante : il suffit de considérer les algorithmes où toutes les configurations sont légitimes, et qui sont entièrement déterminés par les évolutions ci-dessous :



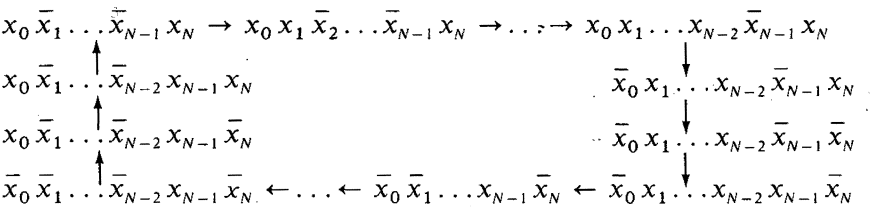
Condition nécessaire : on raisonne par l'absurde. Si on avait une solution pour $N \geqq 4$, alors l'évolution en régime d'exclusion mutuelle serait nécessairement, à une permutation près, de la forme suivante :




Dans ces conditions, en partant de la configuration :

$$x_0 \bar{x}_1 \bar{x}_2 \ldots \bar{x}_{N-1} x_N$$

et en appliquant la séquence régulière d'activations de période $\{1\}, \{2\}, \ldots, \{N-2\}, \{0\}, \{N\}, \{N-1\}$, on aurait l'évolution ci-dessous :




Ceci est contraire à l'auto-stabilisation puisque, dans la configuration $x_0 x_1 \bar{x}_2 \ldots \bar{x}_{N-1} x_N$, $M_0$ et $M_2$ sont privilégiées.

COROLLAIRE 2.2 : *L'algorithme de Dijkstra à trois états sur une boucle est donc optimal pour le nombre d'états.*

ALGORITHME DE DIJKSTRA : Les machines ont pour ensemble d'états, les entiers modulo 3;

machine $M_0$ :

$$\text{si} \quad x[0]+1=x[1] \text{ alors } x[0] := x[0]-1;$$

machine $M_i$, $0<i<N$ :

privilège à gauche :

$$\text{si} \quad x[i-1]=x[i]+1 \text{ alors } x[i] := x[i-1];$$

privilège à droite :

$$\text{si} \quad x[i+1]=x[i]+1 \text{ alors } x[i] := x[i+1];$$

machine $M_N$ :

$$\text{si} \quad (x[0]=x[N-1]) \qquad \text{et} \qquad (x[N-1]+1 \neq x[N])$$
$$\text{alors} \quad x[N] : = x[N-1]$$

COMMENTAIRE : L'analyse de cet algorithme se fait par une méthode analogue à celle du paragraphe 1, en passant par les étapes suivantes :

– il n'y a pas de configuration de blocage et la stabilisation est acquise dès qu'on a une configuration ayant au plus deux composantes;

– entre deux changements d'état de $M_0$, les autres machines ne peuvent changer d'état qu'un nombre fini de fois;

– pour toute séquence régulière d'activations, $M_0$ change d'état une infinité de fois;

– partant d'une configuration à $p$ composantes, $p \geqq 3$, si $M_0$ change d'état deux fois, on obtient une configuration ayant au plus $p-1$ composantes.

On peut montrer que le temps de stabilisation de cet algorithme est supérieur à $n^2+2n-7$.

### 3. CAS DE LA CHAINE

Dans ce paragraphe, nous étudions le nombre minimal d'états nécessaires à l'auto-stabilisation sur une chaîne.

$$M_0 - M_1 - \ldots - M_{i-1} - M_i - M_{i+1} - \ldots - M_N$$

**Figure 3**

PROPOSITION 3.1 : *Sur une chaîne de $N+1$ machines ayant chacune deux états possibles, on peut définir un algorithme d'exclusion mutuelle auto-stabilisant si et seulement si $N=1$.*

*Démonstration :* Condition suffisante : proposition 2.1.

Condition nécessaire : supposons qu'il existe une solution avec $N \geqq 2$. On a alors, en régime d'exclusion mutuelle, une évolution de la forme :

$$x = x_0\, x_1 \ldots x_N \to \bar{x}_0\, x_1 \ldots x_N \to \bar{x}_0\, \bar{x}_1\, x_2 \ldots x_N \to \ldots$$

Comme $M_0$ et $M_2$ doivent, à un instant ultérieur, passer respectivement aux états $x_0$ et $\bar{x}_2$, on a :

$$\delta_0(\bar{x}_0, x_1) = \bar{x}_0 \quad \Rightarrow \quad \delta_0(\bar{x}_0\, \bar{x}_1) = x_0,$$
$$\delta_2(x_2, x_1, x_3) = x_2 \quad \Rightarrow \quad \delta_2(x_2, \bar{x}_1, x_3) = \bar{x}_2.$$

Finalement, dans la configuration $\bar{x}_0 \bar{x}_1 x_2 \ldots x_N$, $M_0$ et $M_2$ sont toutes les deux privilégiées, ce qui est contradictoire.

PROPOSITION 3.2 : *Sur une chaîne de $N+1$ machines dont chacune a trois états possibles, on peut définir un algorithme d'exclusion mutuelle auto-stabilisant tel que, en régime d'exclusion mutuelle, le privilège circule de manière cyclique avec la période :*

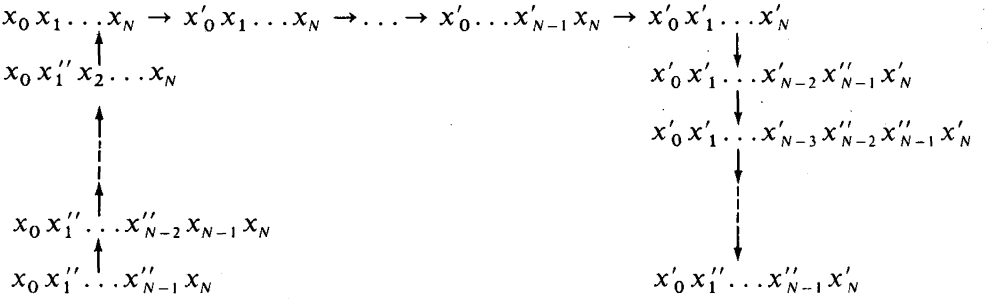
$$0, 1, \ldots, N-1, N, N-1, N-2, \ldots, 1$$

*si et seulement si $N \leqq 2$.*

*Démonstration :* Condition suffisante : le cas $N=1$ est trivial. Si $N=2$, il suffit d'imposer en régime d'exclusion mutuelle, l'évolution ci-dessous, et de la compléter convenablement :

$$\begin{array}{l} x_0 x_1 x_2 \rightarrow x'_0 x_1 x_2 \rightarrow x'_0 x'_1 x_2 \rightarrow x'_0 x'_1 x'_2 \rightarrow x'_0 x''_1 x'_2 \rightarrow x''_0 x''_1 x'_2 \\ \uparrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow \\ x_0 x''_1 x_2 \leftarrow x_0 x''_1 x''_2 \leftarrow x_0 x'_1 x''_2 \leftarrow x''_0 x'_1 x''_2 \leftarrow x''_0 x_1 x''_2 \leftarrow x''_0 x_1 x'_2 \end{array}$$

Condition nécessaire : si l'on avait une solution pour $N \geqq 3$, l'évolution en régime d'exclusion mutuelle serait nécessairement de la forme ci-dessous :




$x_0 x''_1 x''_2 \ldots x''_{N-1} x'_N$ serait alors une configuration de blocage, ce qui contredirait l'auto-stabilisation.

COROLLAIRE : *Si l'on impose qu'en régime d'exclusion mutuelle le privilège circule avec la période :*

$$0, 1, \ldots, N-1, N, N-1, \ldots, 2, 1 \qquad (N \geqq 3),$$

*alors l'algorithme de Dijkstra à quatre états sur une chaîne est optimal pour le nombre d'états.*

ALGORITHME DE DIJKSTRA : La variable d'état de la machine $M_i$ est le couple $(u_i, x_i) \in \{V, F\} \times \{0, 1\}$ :

machine $M_0$ :

On a constamment $u[0] = V$ :

$$\text{si} \quad (u[1] = F) \text{ et } (x[0] = x[1]) \text{ alors } x[0] : = \overline{x[0]};$$

machine $M_i$, $0 < i < N$ :

Privilège à gauche :

```
si  (x[i−1] ≠ x[i]) alors
début  x[i] : = x[i−1];
       u[i] : = V
fin
```

Privilège à droite :

```
si  (x[i] = x[i+1]) et (u[i] = V) et (u[i+1] = F)
alors u[i] : = F
```

machine $M_N$ :

On a constamment $u[N] = F$ :

$$\text{si} \quad x[N] \neq x[N-1] \text{ alors } x[N] : = x[N-1].$$

COMMENTAIRE : La preuve de validité de l'algorithme de Dijkstra se fait par une méthode analogue aux cas précédents :

– il n'y a pas de configuration de blocage;

– entre deux changements d'état de $M_0$, les autres machines ne peuvent changer d'état qu'un nombre fini de fois;

– partant d'une configuration ayant $p$ composantes, $p \geqq 2$, on a avant le second changement d'état de $M_0$, une configuration ayant au plus $p-1$ composantes;

– pour toute séquence régulière d'activations, $M_0$ change d'état une infinité de fois;

– le régime d'exclusion mutuelle est atteint dès qu'on a une configuration où $M_0$ est privilégiée et qui a une seule composante :

$$\begin{matrix} V & F & \ldots & F & F. \\ a & a & & a & a \end{matrix}$$

Le temps de stabilisation de cet algorithme est $N^2 - N$ et, avant qu'on n'atteigne le régime d'exclusion mutuelle, on peut avoir au plus $N^2 - N + 1$ changements d'état.

## 4. ALGORITHMES RÉCURRENTS

Dans ce paragraphe, nous présentons des méthodes permettant de construire, sur un réseau donné, une classe d'algorithmes d'exclusion mutuelle auto-stabilisants définis à partir d'algorithmes associés à des sous-réseaux qui le recouvrent.

DÉFINITION : Soit $Q$, $Q'$ deux ensembles tels que $Q \subset Q'$; deux suites $s = \{q_i\}_{i \geq 1}$, $s' = \{q'_j\}_{j \geq 1}$ de $Q$ et $Q'$ sont dites compatibles si $s$ est la trace de $s'$ sur $Q$, c'est-à-dire si $s$ est la sous-suite de $s'$ correspondant aux $q'_j$ qui appartiennent à $Q$.

PROPOSITION 4.1 : *Soit $G_i = (S_i, \Gamma_i)$, $i = 0, \ldots, n-1$; $n$ réseaux sur lesquels sont définis des algorithmes d'exclusion mutuelle auto-stabilisants et tels que :*

*(a) les $S_i$ ont en commun un sommet unique; la machine placée en ce sommet sera appelée machine centrale;*

*(b) $Q_0 \subseteq Q_1 \subseteq \ldots \subseteq Q_{n-1}$, où $Q_i$ est l'ensemble des états possibles de la machine centrale dans $G_i$;*

*(c) pour tout $q \in Q_i$, les suites des états possibles de la machine centrale à partir de $q$, dans les réseaux $G_j$, $j \geq i$, sont compatibles.*

*Alors :*

*Il existe sur $G = \bigcup_{i=0}^{n} G_i$, un algorithme d'exclusion mutuelle auto-stabilisant où toutes les machines conservent le même ensemble d'états possibles à l'exception de la machine centrale dont la variable d'état devient :*

$$(q, \gamma) \in Q_{n-1} \times \{0, 1, \ldots, n-1\}.$$

DESCRIPTION DE L'ALGORITHME : On convient que, si l'état de la machine centrale est $q \notin Q_i$, alors elle est privilégiée dans le sous-réseau $G_i$ (les $G_i$ sont appelés branches de $G$).

Machine centrale :

*début* : si elle est privilégiée dans $G_\gamma$ relativement à $q$ alors
$\gamma : = \gamma + 1$ modulo $n$;
si elle est privilégiée dans toutes les branches relativement à $q$
alors $q : = \varphi(q)$; (transition prévue dans $G_{n-1}$).
*fin*

Machines voisines de la machine centrale :

si elle est privilégiée dans l'unique branche $G_i$ à laquelle elle appartient et si $\gamma = i$ alors $q.: = \varphi_i(q)$ (transition prévue dans $G_i$).

Autres machines :
aucun changement.

*Preuve de validité :* Les transitions de la machine centrale $M$, par rapport à $q$, et celles des autres machines, respectent les règles associées aux réseaux $G_i$ auxquels elles appartiennent. L'introduction du pointeur $\gamma$ dans la machine centrale a pour unique conséquence de retarder dans certains cas les transitions des machines voisines de la machine centrale.

A tout instant, le sous-réseau $G_\gamma$ évolue comme s'il était autonome; $M$ finit donc par y être privilégiée et à la prochaine activation elle fait $\gamma := \gamma + 1$; on voit donc que, pour toute séquence régulière d'activations, $\gamma$ prend de manière cyclique, les valeurs :

$$0, 1, 2, \ldots, n-1.$$

De part la définition de l'algorithme, il existe un instant où $M$ est privilégiée dans toutes les branches relativement à $q$; à la prochaine activation, elle fait :

$$\gamma := \gamma + 1, \qquad q := \varphi(q).$$

et perd alors le privilège dans toutes les branches. En résumé, on a les propriétés suivantes :

– à chaque instant, la branche $G_\gamma$ détient le privilège et se comporte comme si elle était autonome;

– pour toute séquence régulière d'activations, chaque branche détient le privilège une infinité de fois.

En conséquence, chaque branche finit par se stabiliser; à partir du moment où l'unique privilège de chaque branche sera détenu par la machine centrale, le pointeur $\gamma$ garantira l'exclusion mutuelle entre les branches et donc entre toutes les machines du réseau $G$. L'algorithme est donc auto-stabilisant.

COMMENTAIRE : 1. L'hypothèse de la proposition précédente sur la compatibilité des états assumés par la machine centrale dans les sous-réseaux est assez restrictive. Nous exhibons dans la suite une construction qui supprime cette condition.

2. Lorsque la machine centrale est privilégiée dans toutes les branches relativement à $q$, la transition $\gamma := \gamma + 1$ peut être remplacée par l'affectation à $\gamma$ d'une valeur arbitraire de $\{0, 1, \ldots, n-1\}$.

3. Si $I = \{i : M \text{ privilégiée dans } G_i\}$, $I \neq \{0, 1, \ldots, n-1\}$, la transition $\gamma := \gamma + 1$ peut être remplacée par $\gamma := F(\gamma)$ où $F$ est une fonction quelconque vérifiant :

$$\forall \gamma \in I, \quad F^m(\gamma) \notin I, \qquad \text{avec} \quad m = \text{Card}(I).$$

PROPOSITION 4.2 : *Soit* $G_i=(S_i, \Gamma_i)$, $i=0, 1, \ldots, n-1$, *n réseaux ayant en commun un sommet unique, et sur lesquels sont définis des algorithmes d'exclusion mutuelle auto-stabilisants.*

*Alors :*

*Il existe sur* $G=\bigcup_{i=0}^{n-1} G_i$ *un algorithme d'exclusion mutuelle auto-stabilisant où toutes les machines conservent le même ensemble d'états possibles à l'exception de la machine centrale dont la variable d'état est*

$$(q, \gamma)\in Q_0\times Q_1\times\ldots\times Q_{n-1}\times\{0, 1, \ldots, n-1\},$$

$Q_i$ *ensemble des états de la machine centrale dans* $G_i$.

*Démonstration :* Il suffit de reprendre l'algorithme de la proposition précédente en posant pour la machine centrale :

$$q=(q_0, q_1, \ldots, q_{n-1});$$

une transition relative à $q$ étant maintenant le changement simultané des composantes $q_j$ dans les sous-réseaux $G_j$.

COMMENTAIRE : La proposition que nous venons de démontrer suppose que les réseaux ont un seul sommet en commun. Nous allons maintenant exhiber une construction valable dans le cas où ils ont plus d'un sommet en commun.

PROPOSITION 4.3 : *Soit* $G_i=(S_i, \Gamma_i)$, $i=0, \ldots, n-1$, *n réseaux ayant au moins un sommet en commun et sur lesquels sont définis des algorithmes d'exclusion mutuelle auto-stabilisants.*

*Alors :*

*Il existe sur* $G=\bigcup_{i=0}^{n-1} G_i$ *un algorithme d'exclusion mutuelle auto-stabilisant où toute machine appartenant à un seul de ces réseaux conserve le même ensemble d'états.*

*Démonstration :* Pour une machine $M$, soit :

$$I_M=\{i : M \text{ est dans le réseau } G_i\}=\{i_1, \ldots, i_k\}.$$

On prend, pour ensembles des états de $M$ dans $G$,

$$Q_M=Q_{i_1}\times Q_{i_2}\times\ldots\times Q_{i_k}.$$

Dans le réseau $G$, $M$ peut alors être privilégiée de $k$ manières différentes et totalement indépendantes; tout se passe donc comme si $G$ était une union de $n$ réseaux disjoints.

On choisit ensuite une machine $M_0$ appartenant à tous les réseaux, et on ajoute à sa variable d'état une composante $\gamma$ parcourant $\{0, 1, 2, \ldots, n-1\}$. Tout se passe alors comme si on avait $n$ réseaux ayant une seule machine en commun et on est ramené au contexte de la proposition 4.2.

*Conséquence :* L'application itérée des trois propositions précédentes permet de construire sur un réseau connexe $G$, un algorithme d'exclusion mutuelle auto-stabilisant défini à partir d'algorithmes associés à des sous-réseaux qui le recouvrent.

**Application : algorithme à quatre états sur une chaîne**

Cas de deux machines (rappel).

Machine $M_0$ (type 0) :

$$\text{si} \quad x[0]=x[1] \text{ alors } x[0]:=\overline{x[0]}.$$

Machine $M_1$ (type 1) :

$$\text{si} \quad x[1]\neq x[0] \text{ alors } x[1]:=x[0].$$

Hypothèse de récurrence : supposons le problème résolu sur toute chaîne de longueur $\leqq N$ de sorte que :

- les machines de bout de chaîne aient deux états possibles;
- les autres machines aient quatre états possibles.

En appliquant l'algorithme de la proposition 4.1 à :

$$M_0-M_1-\ldots-M_i,$$
$$M_i-M_{i+1}-\ldots-M_N \qquad (0<i<N).$$

On obtient sur la chaîne d'ordre $N+1$ un algorithme où :

- les machines de bout de chaîne ont deux états possibles;
- les autres ont quatre états possibles.

DESCRIPTION DE L'ALGORITHME : Chaque machine a pour variable d'état :

$$\begin{bmatrix} u[i] \\ x[i] \end{bmatrix}, \qquad u[i]\in\{G, D\}, \quad x[i]\in\{\dot{0}, 1\}.$$

On a constamment $u[0]=D$, $u[N]=G$.

$G$, $D$ signifient gauche et droite respectivement.

Chaque machine $M_i$, $0<i<N$ a par rapport à $x[i]$ :

- un comportement de type 0 à sa droite;
- un comportement de type 1 à sa gauche.

Dans la représentation, on met à gauche de la barre l'état du voisinage avec une flèche sous l'état de la machine considérée, et, à droite de la barre l'état successeur; on adopte la convention $\overline{G}=D$, $\overline{D}=G$.

Machine $M_0$ :

$$\begin{array}{cc|c} D & G & D \\ x & x & \bar{x} \\ \uparrow & & \end{array}$$

Machine $M_i$, $0<i<N$ :

$$\begin{array}{ccc|c} D & u & G & \bar{u} \\ \bar{x} & x & x & \bar{x}; \\ & \uparrow & & \end{array} \qquad \begin{array}{ccc|c} D & G & u & D \\ \bar{x} & x & \bar{x} & x; \\ & \uparrow & & \end{array} \qquad \begin{array}{ccc|c} u & D & G & G \\ x & x & x & x \\ & \uparrow & & \end{array}$$

Machine $M_N$ :

$$\begin{array}{cc|c} D & G & G \\ \bar{x} & x & \bar{x} \\ & \uparrow & \end{array}$$

LEMME 4.4 : (i) *Un privilège qui se déplace vers la droite (resp. gauche) ne peut faire demi-tour que dans deux cas :*

- *réflexion en bout de chaîne;*
- *rencontre de la sous-configuration :*

$$\begin{array}{cc} D & G \\ \alpha & \alpha \end{array} \qquad (resp. \begin{array}{cc} D & G \\ \bar{\alpha} & \alpha \end{array}).$$

(ii) *Un privilège qui se déplace à partir d'une machine du bout de la chaîne laisse derrière elle une sous-configuration sans privilège.*

(iii) *Avant la stabilisation sur une chaîne de $2p$ machines, $M_0$ peut changer d'état au plus $p$-1 fois.*

*Démonstration :* (i) Supposons par exemple que $M_i$ reçoive le privilège de $M_{i-1}$; tout changement d'état de $M_i$ comporte une modification de $u[i]$; si $M_i$ renvoie le privilège à gauche ($u[i] := G$) cela signifie qu'on avait $u[i]=D$, et donc que $M_i$ était privilégiée à droite, d'où la situation annoncée.

(ii) Si par exemple un privilège se déplace à partir de $M_0$, on montre facilement par récurrence que, s'il atteint $M_i$ alors $M_0, M_1, \ldots, M_{i-1}$ ne sont pas privilégiés.

(iii) Si la stabilisation n'est pas encore réalisée, on a au moins deux privilèges dans le réseau; pendant ce temps, le privilège lié à tout changement d'état de $M_0$

doit donc se réfléchir sur une sous-configuration de la forme indiquée dans (i) et on vérifie que ceci peut se produire $p$-1 fois au plus.

PROPOSITION 4.5 : *Si $\varphi(N)$ et $\psi(N)$ désignent respectivement le temps de stabilisation et le nombre maximal de changements d'état avant la stabilisation, sur une chaîne de N machines, alors :*

$$\psi(2p-1)\leqq\psi(2p)=2(p-1)^2+1; \qquad \varphi(2p-1)\leqq\varphi(2p)=2(p-1)^2.$$

*Démonstration :* On vérifie aisément que :

$$\psi(3)=\varphi(3)=0; \qquad \varphi(4)=2; \qquad \psi(4)=3.$$

Supposons le résultat vrai pour $p$ et considérons une chaîne d'ordre $2p+2$, on a :

$$S' : \quad M_0-M_1-M_2,$$

$$S'' : \quad M_2-M_3-\ldots-M_{2p+1}-M_{2p+2},$$

$$S=S'\cup S''.$$

Un privilège d'une machine $M_i$ est dit de type $S'$ (resp. $S''$) si :

$$i<2 \qquad (\text{resp. } i>2)$$

ou :

$$i=2 \qquad \text{et} \qquad u[2]=G \quad (\text{resp. } u[2]=D).$$

Si $u[2]=G$ (resp. $u[2]=D$) alors un privilège de type $S'$ devient après au plus 4 (resp. 2) changements d'état dans $S'$, un privilège de type $S''$.

Seule la machine $M_2$ peut transformer un privilège de type $S''$ en un privilège de type $S'$ et, avant le régime d'exclusion mutuelle, ceci peut, d'après le lemme 4.4 se produire au plus $p$-2 fois (resp. $p$-1 fois) si au départ $u[2]=G$ (resp. $u[2]=D$).

En conséquence, on a :

$$\psi(2p+2)\leqq\max\{4+4(p-2)+\psi(2p),\ 2+4(p-1)+\psi(2p)\}\leqq 2p^2+1,$$

$$\varphi(2p+2)\leqq\max\{4+4(p-2)+\varphi(2p),\ 2+4(p-1)+\varphi(2p)\}\leqq 2p^2.$$

Ces bornes sont atteintes comme suit : on part de la configuration :

$$\begin{array}{cccccccccccc} D & G & D & G & D & G & \ldots & D & G & D & G \\ \alpha & \alpha & \bar{\alpha} & \bar{\alpha} & \alpha & \alpha & \ldots & \theta & \theta & \bar{\theta} & \theta \end{array}$$

D'abord, pour $i=1, 2, \ldots, p-1$ on procède à l'activation de 0, 1, 2, $\ldots$, $2i-1$, $2i$, $2i-1$, $2i-2$, $\ldots$, 1 dans cet ordre.

Ensuite on active :

$$0, 1, 2, \ldots, 2p-3, 2p-2 \text{ dans cet ordre.}$$

Enfin on active simultanément $2p-1$ et $2p+1$.

*Exemple :*

$$\begin{array}{cccccccc} D & G & D & G & D & G & D & G \\ \alpha & \alpha & \bar{\alpha} & \bar{\alpha} & \alpha & \alpha & \bar{\alpha} & \alpha \end{array}$$

$$\begin{array}{ccccccc} D & D & G & . & . & . & . \\ \bar{\alpha} & \alpha & \alpha & . & . & . & . \end{array}$$

$$\begin{array}{ccccccc} . & G & . & . & . & . & . \\ . & \bar{\alpha} & . & . & . & . & . \end{array}$$

$$\begin{array}{ccccccc} D & D & D & D & G & . & . \\ \alpha & \bar{\alpha} & \alpha & \bar{\alpha} & \bar{\alpha} & . & . \end{array}$$

$$\begin{array}{ccccccc} . & G & G & G & . & . & . \\ . & \alpha & \bar{\alpha} & \alpha & . & . & . \end{array}$$

$$\begin{array}{ccccccc} D & G & D & D & D & D & . & G \\ \bar{\alpha} & \alpha & \bar{\alpha} & \alpha & \bar{\alpha} & \alpha & . & \bar{\alpha} \end{array}$$

Nous ne mettons que les états des machines activées.

COMMENTAIRE : Pour l'algorithme de Dijkstra à quatre états sur une chaîne, on a :

$$\psi(2p)=2(p-1)(2p-1)+1, \qquad \varphi(2p)=2(p-1)(2p-1).$$

L'algorithme récurrent se stabilise donc deux fois plus vite. Ce phénomène se justifie par le fait qu'il se déduit de celui de Dijkstra par suppression, de certains privilèges, ce qui, contrairement aux solutions de [1], lui permet de toujours garantir l'exclusion mutuelle entre deux machines voisines.

**Sur la minimisation du nombre d'états**

Soit un réseau de $N+1$ machines reliées entre elles selon un arbre $H$; la méthode récurrente permet d'y définir un algorithme où la machine située au sommet $i$ a une variable d'état :

$$(x, \gamma) \in \{0, 1\} \times \{0, 1, \ldots, n-1\}, \qquad n=d_H(i) \text{ (degré du sommet } i).$$

L'ensemble des états possibles du réseau est donc :

$$2^{N+1} \times \prod_{i=0}^{N} d_H(i).$$

Dans le cas d'un réseau connexe quelconque, on peut donc aborder le problème de la détermination d'un algorithme ayant peu de variables d'état en recherchant un arbre $H$ qui recouvre le réseau et tel que le produit des degrés de ses sommets est minimal; ce problème combinatoire est $NP$-complet [5].

## CONCLUSION

L'exclusion mutuelle dans un système réparti est un cadre excellent pour l'illustration de certaines caractéristiques propres aux organisations de type cellulaire.

D'abord la seule existence de la solution montre qu'on peut réaliser un comportement complexe, l'exclusion mutuelle sur un grand nombre de machines, à partir d'algorithmes locaux très simples où chaque machine ne communique qu'avec un très faible nombre de voisines.

Ensuite elle montre l'aptitude de ces structures à l'auto-régulation; en effet l'auto-stabilisation signifie que, si à la suite du mauvais ajustement de la variable d'état d'une machine, l'exclusion mutuelle est mise en défaut, alors au bout d'un temps fini, le système revient de lui-même à un état de bon fonctionnement, c'est-à-dire d'exclusion mutuelle.

Enfin, elle permet de montrer la grande flexibilité de ces structures; cet aspect a été étudié dans [2]; plus précisément, il y est expliqué comment on peut organiser le système de manière à ce que chaque machine puisse quitter le réseau ou au contraire s'y insérer, sans compromettre le bon fonctionnement de l'ensemble.

L'approche récurrente permet d'obtenir une grande classe d'algorithmes qui se stabilisent vite et dont le mécanisme est facile à comprendre; elle permet d'obtenir le théorème d'existence de solution pour un réseau connexe quelconque à partir du cas trivial associé à un couple de machines.

## REMERCIEMENTS

Nous exprimons notre profonde reconnaissance aux Professeurs N. Gastinel et J. P. Verjus dont les nombreuses remarques et suggestions ont été très précieuses au cours de l'élaboration de ce travail.

## BIBLIOGRAPHIE

1. E. W. Dijkstra, *Self-Stabilizing Systems in Spite of Distributed Control*, Comm. A.C.M., vol. 17, n° 11, novembre 1974, p. 643-644.
2. J. Mossière, J. P. Verjus et M. Tchuente, *Mutual Exclusion in Computer Networks*, Rapport interne, I.M.A.G., juin 1977, Grenoble.

3. F. ROMANI, *Cellular Automata Synchronization*, Information Sciences, vol. 10, 1976, p. 299-318.
4. P. ROSENSTIEHL, J. R. FIKSEL et A. HOLLIGER, *Intelligent Graphs: Networks of Finite Automata Capable of Solving Graph Problems* dans Graph Theory and Computing, R. C. READ, éd., Academic Press, 1973, p. 219-265.
5. M. TCHUENTE, *On the Complexity of a Degree Constrained Spanning Tree Problem.* Rapport de Recherche n° 190, I.M.A.G., février 1980, Grenoble.