Protocoles auto-stabilisants synchrones d'exclusion mutuelle pour les anneaux anonymes et uniformes

Philippe Duchon [†]— Nicolas Hanusse[‡]— Sébastien Tixeuil [§]

LaBRI, Université Bordeaux I, 351 Cours de la libération, 33400 Talence. `duchon,hanusse@labri.fr`
LRI – CNRS UMR 8623 et INRIA Projet Grand Large, Université Paris-Sud XI. `tixeuil@lri.fr`

Nous proposons plusieurs protocoles auto-stabilisants pour les réseaux synchrones anonymes et uniformes en anneau unidirectionel où les processeurs communiquent par échange de messages. Dans le cas où la taille de l'anneau n est inconnue, nous améliorons d'un facteur n le meilleur temps de service connu (en conservant les meilleures complexités connues pour le temps de stabilisation et la mémoire utilisée). Quand la taille de l'anneau est connue, nous présentons un protocole optimal en mémoire (constante et indépendante de n), temps de stabilisation et temps de service (tous deux de l'ordre de n).

Keywords: algorithmique distribuée, exclusion mutuelle, chaîne de Markov, temps de stabilisation, temps de service

1 Introduction

Le problème de l'exclusion mutuelle est un problème fondamental dans le domaine de l'informatique répartie. Considérons un système réparti de n processeurs. Tous les processeurs, de temps à autre, peuvent avoir a exécuter une section critique de leur code durant laquelle exactement un processeur est autorisé à utiliser une ressource partagée. Un système réparti qui résoud le problème de l'exclusion mutuelle doit garantir les deux propriétés suivantes : *(i) exclusion mutuelle* : exactement un processeur est autorisé à exécuter sa section critique à un instant donné ; *(ii) équité* : tout processeur doit être en mesure d'exécuter sa section critique infiniment souvent au cours de l'exécution.

Une technique classique consiste à faire passer à chaque processeur un message spécial appelé *jeton*. La réception du jeton signifie que le processeur peut entrer en section critique. Ce jeton doit permettre le respect des contraintes de l'exclusion mutuelle : être présent en un unique exemplaire et passer infiniment souvent par chaque processeur.

Le concept de l'auto-stabilisation a été introduit pour la première fois par Dijkstra en 1974 [Dij74]. Ce concept est maintenant considéré comme la technique la plus générale pour concevoir des systèmes qui tolèrent des défaillances transitoires arbitraires. Un système auto-stabilisant garantit que, en partant d'un état initial arbitraire, le système converge vers un état correct en un nombre fini d'étapes, et reste correct en l'absence de nouvelles défaillances (cf. [Dol00]).

Intuitivement, un protocole d'exclusion mutuelle auto-stabilisant par passage de jeton garantit que, même si on part d'un état où la spécification de l'exclusion mutuelle n'est pas respectée (zéro ou plusieurs jetons sont présents dans le système), alors en un nombre fini d'étapes, un seul jeton circule équitablement dans le réseau. En pratique, on se borne à prouver qu'à partir d'une configuration à plusieurs jetons, on aboutit en un temps fini à une configuration à jeton unique. En effet, [KP93] ont prouvé que dans le cas où on se trouve dans un système réparti où les processeurs communiquent par passage de messages, il est indispensable de disposer d'un mécanisme de temporisation (*timeout*) pour injecter des jetons spontanément : si un tel

[†] financé partiellement par EC's IHRP Programme, grant HPRN-CT-2001-00272, "Algebraic Combinatorics in Europe"
[‡] financé par le JemSTIC MOBICoop
[§] financé par le JemSTIC MOBICoop

mécanisme n'est pas disponible, le système ne peut être auto-stabilisant puisqu'il se retrouverait bloqué en démarrant d'une configuration initiale sans messages.

1.1 Travaux précédents

Un réseau est *uniforme* si tout processeur exécute le même code, et il est *anonyme* si les processeurs ne disposent par d'identificateurs pour exécuter des sections de code différentes. Si un protocole fonctionne dans un réseau uniforme et anonyme, alors il fonctionne *a fortiori* dans un réseau non uniforme ou non anonyme. Depuis les trois protocoles d'exclusion mutuelle auto-stabiliante par passage de jeton proposés dans l'article fondateur [Dij74], de nombreux travaux ont traité de ce problème dans différents contextes et par exemple [Her90, BCD95, BGJ99, DGT00, Ros00, Joh02] dans le cas des anneaux de processeurs unidirectionnels anonymes et uniformes.

Un protocole d'exclusion mutuelle par passage de jeton est *transparent* vis à vis de l'application qui utilise le protocole si il ne modifie pas le format des jetons qui sont échangés par l'application. Une telle propriété est souhaitable si par exemple le contenu du jeton est utilisé par l'application (c'est le cas dans un réseau de type *Token Ring* ou *FDDI*, où le jeton contient également les informations devant être transmises au destinataire). En effet, un protocole transparent est plus facile à implanter (il ne modifie pas le format des trames de l'application) et plus facile à intégrer à des réseaux hétérogènes (dont certaines parties utilisent un protocole de passage de jeton différent). En outre, la charge de la vérification de l'intégrité des messages peut être déléguée entièrement à l'application. Parmi les protocoles précités, seuls [Her90, BCD95] et le protocole synchrone de [DGT00] sont transparents vis à vis de l'application qui utilise le protocole. Les protocoles présentés dans [BGJ99, DGT00, Ros00, Joh02] utilisent soit plusieurs types de jetons (et donc ajoutent un champ *type* aux messages de l'application), soit des informations supplémentaires à chaque jeton circulant de manière à assurer la stabilisation.

Un critère important pour evaluer l'efficacité de la stabilisation d'un protocole est de calculer son *temps de stabilisation* ou *temps de convergence*, noté T, c'est à dire le temps qu'il faut pour passer d'une configuration contenant un nombre arbitraire de jetons à une configuration à jeton unique. Comme montré dans [BP89], il est impossible de résoudre le problème de l'exclusion mutuelle auto-stabilisante dans un anneau unidirectionnel anonyme et uniforme au moyen d'un protocole déterministe. Aussi les solutions précédentes sont elles toutes probabilistes. Parmi celles-ci, [Her90, BCD95] ne proposent pas de calcul du temps de stabilisation, et les temps de stabilisation moyens attendus de [BGJ99, DGT00, Joh02] sont de l'ordre de n^3, où n désigne la taille de l'anneau, et celui de [Ros00] est de l'ordre de n^2. Il est évident que le temps de stabilisation est en $\Omega(n)$, puisque si on part d'une configuration à deux jetons diamétralement opposés, il faut au moins $n/2$ unités de temps pour qu'un jeton rattrape l'autre.

Un autre critère d'évaluation est celui du *temps de service*, c'est à dire le temps, en phase stabilisée, entre deux passages du jeton sur un processeur. Ce temps est important pour évaluer les performances du protocole quand il n'y a pas de défaillances et ainsi évaluer son surcoût par rapport à un protocole non stabilisant. Dans un système à n processeurs, le temps de service est en $\Omega(n)$, puisque si chaque processeur attend le minimum de temps, il attend autant que les autres. Le temps de service n'est pas calculé dans [Her90, BCD95, BGJ99], et il est dans [DGT00, Ros00, Joh02] de l'ordre de n^3, n^2, et n, respectivement.

1.2 Notre contribution

Nous proposons plusieurs protocoles auto-stabilisants pour les réseaux synchrones anonymes et uniformes en anneau unidirectionel où les processeurs communiquent par échange de messages. Les deux premiers protocoles sont des transpositions dans un modèle à passage de messages des protocoles de [Her90, BCD95] (pour le protocole **SansMémoire**) et de [DGT00] (pour le protocole **Mémoire1bit**) qui utilisent un modèle à mémoire partagée. Par une analyse fine de complexité, nous montrons que l'équivalent de [Her90, BCD95] converge en temps $\Theta(n^2)$ en moyenne, et que son temps de service est en $\Theta(n^2)$ en moyenne (ces deux complexités n'étaient pas calculées dans les articles cités). Puis nous montrons que l'équivalent du protocole synchrone de [DGT00] converge en temps moyen $\Theta(n^2)$ et que son temps de service est en $\Theta(n)$ (il était prouvé seulement $O(n^2)$ dans l'article original). Ensuite, nous proposons plusieurs protocoles originaux basés sur la notion de *ralentisseur*. Chaque processeur peut se proclamer ralentisseur et ralentir les jetons qu'il reçoît avec une certaine probabilité. Suivant la puissance du ralentisseur considéré (quantité de mémoire disponible), les résultats de complexités obtenus sont différents, mais chacun des protocoles que nous proposons a un temps de convergence moyen, noté $\mathbb{E}(T)$, et un temps de service moyen en $O(n)$ (de manière certaine pour **Mémoire1bit**).

Protocole	Connaissance de n	Temps de stabilisation	Temps de service	Mémoire	Uniforme	Transparent
[BGJ99]	oui	$\Theta(n^3)$	$O(n^3)$	$O(\log(n))$	oui	non
[Ros00]	non	$\Theta(n^2)$	$O(n^2)$	$O(1)$	oui	non
[Joh02]	oui	$O(n^3)$	$O(n)$	$O(\log(n))$	oui	non
SansMémoire	non	$\Theta(n^2)$	$\Theta(n)$	0	oui	oui
Mémoire1bit	non	$\Theta(n^2)$	$\Theta(n)$	$O(1)$	oui	oui
Ralentisseur1	oui	$\Theta(n)$	$\Theta(n)$	$O(\log(n))$	oui	oui
Ralentisseur2	oui	$\Theta(n)$	$\Theta(n)$	$O(1)$	oui	oui

FIG. 1: Récapitulatif des résultats

Tous les protocoles uniformes que nous présentons (cf. figure 1) sont transparents vis à vis de l'application qui utilise l'exclusion mutuelle. Quand la taille de l'anneau n'est pas connue, le protocole **Mémoire1bit** est le plus intéressant, et quand elle est connue, le protocole **Ralentisseur2** est le plus intéressant.

2 Préliminaires

2.1 Modèle d'exécution

Dans cet article, nous considérons un système synchrone où tous les processeurs agissent à chaque pulsation de l'horloge globale. Si les processeurs sont synchronisés, ils n'ont cependant pas accès à un temps absolu, mais seulement à des pulsations de temps. Une telle phase où tous les processeurs agissent est apellée une *action* du système (désignée par la suite sous le terme de *ronde*). On suppose également que chaque processeur peut interroger la couche de communication pour savoir si un message (par exemple un jeton) lui a été délivré.

2.2 Rappel sur les chaînes de Markov

Nous utilisons la terminologie de Norris [Nor97] sur les chaînes de Markov et les marches aléatoires. Ces hypothèses classiques peuvent être utilisées car le réseau a un comportement synchrone ; pour l'utilisation des chaînes de Markov dans un environnement asynchrone, cf. [DFP01]. Soit $P_{n\times n}$ une matrice stochastique, c'est-à-dire que la somme sur chaque ligne est égale à 1. Une *chaîne de Markov à temps discret*, notée $(X_t)_{t\leq 0}$ sur un ensemble d'états X est une séquence de variables aléatoires $X_0, X_1, \ldots$ où $X_i \in X$ et telle que X_{i+1} ne dépend que de X_i et $\mathbb{P}r(X_{i+1} = x | X_i = y) = p_{x,y}$. La matrice P est appelée la *matrice de probabilité de transition*.

Un nœud x *mène* à un nœud y si $\mathbb{P}r(X_j = y$ pour $j \geq i | X_i = x)$. Un état y est dit *absorbant* si y ne mène à aucun autre état. Le *temps d'accés moyen* ou *temps d'accés* $\mathbb{E}_x^y$ est le nombre moyen d'étapes en partant d'un nœud x pour atteindre un nœud y.

Analogie temps d'accés et temps de stabilisation : si $x = 0$ et y désigne la distance initiale entre deux jetons consécutifs, le temps d'accés représente le temps au bout duquel la distance devient nulle, *i.e.* les deux jetons fusionnent. Nous utilisons un théorème bien connu dans le domaine des chaînes de Markov pour calculer le temps d'accés :

Theorème 1 *Le vecteur des temps d'accés $\mathbb{E}^t = (\mathbb{E}_x^t : x \in V)$ est la solution non-négative minimale du système d'équations linéaires :*

$$\begin{cases} \mathbb{E}_t^t = 0 \\ \mathbb{E}_x^t = 1 + \sum_{y\neq t} p_{x,y}\mathbb{E}_y^t \ \textit{pour}\ x \in V \end{cases}$$

ainsi que le lemme suivant :

Lemme 2 (cf. page 5 [Nor97]) *Soit une chaîne de Markov (X_t) à deux états $\{1,2\}$ de matrice de transition P de la forme : $P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$ alors*

$$\mathbb{P}r(X_t = 1 | X_0 = 1) = \begin{cases} \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^t & \text{pour } \alpha+\beta > 0 \\ 1 & \text{pour } \alpha+\beta = 0 \end{cases}$$

Pour analyser nos algorithmes, il est nécessaire de connaître les probabilités que la distance diminue, augmente ou reste inchangée, ce qui n'est pas toujours aisé.

3 Algorithmes

Le Protocole distribué que nous proposons revient à une marche aléatoire des jetons dans l'anneau, et peut être vu comme une transposition dans le modèle à passage de messages des protocoles de [Her90, BCD95] :

Protocole SansMemoire Chaque processeur i exécute le code suivant : à chaque unité de temps, si un jeton a été délivré à i, avec probabilité p, i retransmet le jeton au processeur suivant, et garde le jeton délivré avec probabilité $1-p$.

Bien qu'il soit connu que le temps moyen pour effectuer le tour d'un anneau (qu'il soit orienté ou non) soit d'ordre $\Theta(n^2)$ [Bol85], nous montrons que le temps moyen de convergence pour un nombre quelconque de jetons est du même ordre. Pour cela, nous utilisons le lemme suivant :

Lemme 3 *Soit C_d une chaîne à $d+1$ états $0,1,\ldots,d$ et $q \in]0,1/2]$. Si l'état 0 est absorbant et si les probabilités de transition sont telles que :*

$$\begin{cases} p_{0,i} = 0 \text{ pour } i \neq 0 \\ p_{i,i-1} = q \text{ pour } i \geq 1 \\ p_{i,i} = 1-2q \text{ pour } 1 \leq i \leq d-1 \\ p_{d,d} = 1-q \\ p_{i,i+1} = q \text{ pour } 1 \leq i \leq d-1 \end{cases}$$

alors le temps d'accés à l'état 0 en partant de i est $\mathbb{E}_i^0 = \frac{i}{2q}(2d-i+1)$.

Theorème 4 *Soit un anneau de taille n comportant un nombre quelconque de jetons dans son état initial. Sans mémoire et sans connaissance de n, le temps moyen de convergence du protocole **SansMemoire** est $\frac{n^2}{8p(1-p)} < \mathbb{E}(T) < \frac{n^2}{2p(1-p)}(\frac{\pi^2}{6}-1) + \frac{n\log n}{p(1-p)}$. Pour $p \in]0..1[$ constant, $\mathbb{E}(T) = \Theta(n^2)$.*

Remarque 5 *Le temps moyen de convergence est minimal pour $p = 1/2$. L'étude précise du protocole **SansMémoire** montre que le temps de convergence ne dépend que très faiblement du nombre initial de jetons : pour n assez grand et $p = 1/2$, $\mathbb{E}(T) > n^2/2$ pour deux jetons et $\mathbb{E}(T) < 1.3n^2$ pour n jetons.*

Le temps de service moyen est aussi d'ordre $\Theta(n^2)$. Le Protocole suivant, nommé **Memoire1bit**, assure un temps de service 2-optimal.

Protocole Mémoire1bit Chaque processeur i exécute le code suivant : à chaque unité de temps, si un jeton vient d'être délivré à i, alors i exécute le protocole **SansMémoire** pendant une ronde. Si un jeton a été délivré à i à la ronde précédente, alors i retransmet le jeton au processeur suivant. Le processeur i a besoin de seulement un bit de mémoire pour savoir s'il a reçu un message à la ronde précédente.

L'analyse de cet protocole mène au résultat suivant :

Theorème 6 *Soit un anneau de taille n comportant un nombre quelconque de jetons dans son état initial. Avec 1 bit de mémoire et sans connaissance de n, le temps de service du protocole **Memoire1bit** est compris entre n et $2n$ et garantit que $\frac{2n^2}{5} < \mathbb{E}(T) < \frac{32n^2}{3}(\frac{\pi^2}{6}-1) + \frac{64n\log n}{3}$ avec $p = 1/2$.*

Introduisons la possibilité que des processeurs particuliers aient un comportement différent des autres processeurs. Nous appelons ces processeurs des *ralentisseurs* : ils ne font pas avancer les jetons (de manière certaine). Les autres processeurs (qui ne sont pas des ralentisseurs) font avancer les jetons (de manière certaine également).

Protocole Ralentisseur1 Chaque processeur i exécute le code suivant : toutes les n unités de temps, le processeur i lance un dé pour savoir s'il doit se comporter en tant que ralentisseur ou pas. Il est ralentisseur avec probabilité $1/2n$.

Theorème 7 *Soit un anneau de longueur n comportant un nombre quelconque de jetons dans son état initial. Si le nombre de processeurs est connu alors le protocole* ***Ralentisseur1*** *a un temps de convergence moyen majoré par* $(1+3e^{3/2})n$.

Pour être en mesure de compter n unités de temps, le protocole **Ralentisseur1** nécessite une mémoire de taille logarithmique. Le Protocole **Ralentisseur2** n'a pas besoin de cet espace mémoire, et n'utilise qu'une mémoire constante.

Protocole Ralentisseur2 Chaque processeur i exécute le code suivant : à chaque unité de temps, si l'état du processeur i est normal, il devient ralentisseur avec probabilité $q = \frac{1}{n(n-1)}$; sinon (s'il est ralentisseur), il devient normal avec probabilité $p = 1/n$.

Theorème 8 *Soit un anneau de longueur n comportant un nombre quelconque de jetons dans son état initial. Si le nombre de processeurs est connu alors le protocole* ***Ralentisseur2*** *a un temps moyen de convergence* $\mathbb{E}(T) \leq n(1+\frac{e^4}{e-1})$ *et ne nécessite qu'une mémoire de taille constante.*

Références

[BCD95] J. Beauquier, S. Cordier, and S. Delaët. Optimum probabilistic self-stabilization on uniform rings. In *WSS95 Second Workshop on Self-Stabilizing Systems*, pages 15.1–15.15, 1995.

[BGJ99] J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *PODC99 18th Annual ACM Symposium on Principles of Distributed Computing*, pages 199–208, 1999.

[Bol85] B. Bollobas. *Random Graphs*. Academic Press, London, 1985.

[BP89] J. E. Burns and J. Pachl. Uniform self-stabilizing rings. *ACM Transactions on Programming Languages and Systems*, 11(2) :330–344, 1989.

[DFP01] M. Duflot, L. Fribourg, and C. Picaronny. Finite-state distributed algorithms as markov chains. In *DISC01 Distributed Computing 15th International Symposium, Springer LNCS :2180*, pages 240–255, 2001.

[DGT00] A. K. Datta, M. Gradinariu, and S. Tixeuil. Self-stabilizing mutual exclusion using unfair distributed scheduler. In *IPDPS00 14th International Parallel and Distributed Processing Symposium*, pages 465–470, 2000.

[Dij74] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11) :643–644, 1974.

[Dol00] S. Dolev. *Self-stabilization*. The MIT Press, 2000.

[Her90] T. Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2) :63–67, 1990.

[Joh02] C. Johnen. Service time optimal self-stabilizing token circulation protocol on anonymous unidrectional rings. In *SRDS02 21th Symposium on Reliable Distributed Systems*, pages 80–89. IEEE Computer Society Press, 2002.

[KP93] S. Katz and K. J. Perry. Self-stabilizing extensions for message-passing systems. *Distributed Computing*, 7(1) :17–26, 1993.

[Nor97] J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.

[Ros00] L. Rosaz. Self-stabilizing token circulation on asynchronous uniform unidirectional rings. In *PODC00 19th Annual ACM Symposium on Principles of Distributed Computing*, pages 249–258, 2000.