

Élection Autostabilisante dans les Réseaux à Haute Dynamicité

Karine Altisen, Stéphane Devismes, Anaïs Durand, Colette Johnen, Franck Petit

▶ **To cite this version:**

Karine Altisen, Stéphane Devismes, Anaïs Durand, Colette Johnen, Franck Petit. Élection Autostabilisante dans les Réseaux à Haute Dynamicité. ALGOTEL 2020 – 22èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Sep 2020, Lyon, France. hal-02791667

HAL Id: hal-02791667
https://hal.science/hal-02791667

Submitted on 5 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Élection Autostabilisante dans les Réseaux à Haute Dynamicité

Karine Altisen[1], Stéphane Devismes[1], Anaïs Durand[2], Colette Johnen[3] et Franck Petit[4]

[1]*VERIMAG, Université Grenoble Alpes, Grenoble-INP, Grenoble, France*
[2]*LIMOS, Université Clermont Auvergne, Clermont-Ferrand, France*
[3]*LaBRI, Université de Bordeaux, Bordeaux, France*
[4]*LIP6, Sorbonne Université, Paris, France*

Nous nous intéressons à la conception d'algorithmes autostabilisants pour des réseaux identifiés hautement dynamiques. Précisément, nous considérons le problème de l'élection dans trois classes de graphes dynamiques (TVG) : la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$ des TVG de diamètre temporel borné par Δ, la classe $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ des TVG de diamètre temporel quasiment borné par Δ et la classe $\mathcal{TC}^{\mathcal{R}}$ des TVG à connectivité temporelle récurrente. Nous montrons qu'en dépit des identités, dans les classes $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ et $\mathcal{TC}^{\mathcal{R}}$, tout algorithme autostabilisant d'élection nécessite la connaissance exacte du nombre de processus. Puis nous proposons trois algorithmes d'élection. Le premier, pour la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$, stabilise en au plus 3Δ rondes. Dans $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ et $\mathcal{TC}^{\mathcal{R}}$, le temps de stabilisation d'un algorithme autostabilisant d'élection ne peut pas être borné. Cependant, nous montrons que nos deux solutions sont spéculatives, c'est-à-dire qu'elles ont de bonnes performances dans des cas favorables ; en effet, elles stabilisent en $O(\Delta)$ rondes lorsque l'on se restreint à la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$.

Mots-clefs : Autostabilisation, graphe dynamique, élection de leader.

1 Introduction

À partir d'une configuration initiale quelconque, un système autostabilisant retrouve en temps fini une configuration dite *légitime* à partir de laquelle son comportement est conforme à sa spécification. Une telle initialisation peut être vue comme la résultante d'un nombre fini de fautes transitoires dans le système. Ainsi, l'*autostabilisation* est une propriété générale caractérisant l'aptitude d'un système distribué à tolérer des *fautes transitoires*. Jusqu'à présent la plupart des travaux sur l'autostabilisation se sont concentrés sur les réseaux à topologie fixe. Nous considérons ici des systèmes *dynamiques*, c'est-à-dire des réseaux dans lesquels des changements topologiques peuvent survenir (*e.g.*, l'ajout ou la suppression de liens de communication). Lorsque les changements topologiques sont détectables (en temps fini et localement par les processus) et de fréquence suffisamment faible, ces évènements peuvent être considérés comme transitoires. Dans ce cas, le temps entre deux périodes de changements topologiques est supposé suffisamment grand pour permettre à un algorithme autostabilisant de récupérer puis d'avoir un comportement correct pendant une période assez longue avant que d'autres changement topologiques ne surviennent. Ainsi, l'autostabilisation reste une approche intéressante pour traiter ce type de problème. Il faut d'ailleurs noter que des variantes de l'autostabilisation comme la *superstabilisation* [4] ou la *stabilisation progressive* [1] permettent de traiter spécifiquement et efficacement les changements topologiques transitoires.

Ici, nous étudions des *réseaux à haute dynamicité*, c'est-à-dire, des réseaux où la fréquence des changements topologiques est trop élevée pour les considérer comme des évènements transitoires. Dans ce type de système, les changements topologiques ne doivent plus être considérés comme des fautes mais comme faisant partie intégrante du système. Les approches précédentes sont totalement inopérantes dans ce cas.

Nous représentons la dynamicité du réseau en utilisant les *graphes évoluant dans le temps* [3] (*TVG* pour *Time-Varying Graphs*). Un TVG est représenté à l'aide d'un graphe $G = (V,E)$, d'un intervalle $\mathcal{T}$ sur $\mathbb{N}^*$

(exprimant sa durée de vie) et d'une fonction de présence $\rho : E \times \mathcal{T} \to \{0,1\}$ telle que $\rho(e,t) = 1$ si et seulement si l'arête e existe au temps t dans le réseau. Les TVG sont généralement regroupés en classes caractérisées par des propriétés temporelles. Nous considérons ici trois classes fondamentales : la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$ des TVG de diamètre temporel borné par Δ, la classe $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ des TVG de diamètre temporel quasiment borné par Δ et la classe $\mathcal{TC}^{\mathcal{R}}$ des TVG à connectivité temporelle récurrente. Noter que par définition, on a $\mathcal{TC}^{\mathcal{B}}(\Delta) \subseteq \mathcal{TC}^{\mathcal{Q}}(\Delta) \subseteq \mathcal{TC}^{\mathcal{R}}$. Notre but est de proposer des algorithmes autostabilisants efficaces d'élection pour chacune de ces classes.

Contributions. Nous nous sommes tout d'abord intéressés aux conditions pour lesquelles notre problème pouvait être résolu. Nos résultats font apparaître que la connaissance exacte du nombre total de processus n est souvent indispensable. Précisément, bien que les processus soient identifiés, nous montrons que la connaissance exacte de n est nécessaire pour résoudre l'élection autostabilisante dans les classes $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ et $\mathcal{TC}^{\mathcal{R}}$. Pour obtenir ces deux résultats, nous avons formalisé la propriété de *taille-ambiguité* qui représente le fait d'« avoir une connaissance partielle de n », comme par exemple en connaître une borne supérieure. Informellement, un algorithme s'exécutant dans un système donné est *taille-ambigu* s'il existe des sous-ensembles (stricts) de ces processus ne partageant pas suffisamment de connaissances initiales sur n pour leur permettre de déceler (a priori) que le système ne se restreint pas seulement à eux.

Nous avons ensuite proposé des algorithmes d'élection autostabilisants pour les trois classes ci-dessus. Précisément, nous proposons un algorithme pour la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$ où les processus connaissent Δ mais pas n. Cet algorithme montre ainsi que nos résultats sont précis : aucune connaissance de n n'est nécessaire pour résoudre l'élection autostabilisante dans $\mathcal{TC}^{\mathcal{B}}(\Delta)$, alors qu'au contraire la connaissance exacte de n est nécessaire dans $\mathcal{TC}^{\mathcal{Q}}(\Delta)$. Nous proposons ensuite une solution pour $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ où les processus connaissent Δ et n et une solution pour $\mathcal{TC}^{\mathcal{R}}$ où les processus connaissent n mais nécessite une mémoire non-bornée.

Nous montrons que le temps de stabilisation de notre solution pour $\mathcal{TC}^{\mathcal{B}}(\Delta)$ est asymptotiquement optimal, c'est-à-dire en $O(\Delta)$ rondes. Au contraire, par définition de $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ et $\mathcal{TC}^{\mathcal{R}}$, le temps de stabilisation d'un algorithme autostabilisant ne peut être borné en général dans ces classes (sauf pour des spécifications triviales). Cependant nous montrons que nos solutions sont spéculatives. Un système est *spéculatif* [6] si toutes ses exécutions satisfont la spécification attendue, et qu'il existe un sous-ensemble intéressant d'exécutions où les performances sont significativement meilleures. L'idée générale derrière cette notion est que les pires cas sont souvent rares et qu'il est intéressant d'évaluer les performances en excluant de tels scénarios. Il faut noter que Dubois et Guerraoui [5] ont introduit la notion de spéculation en autostabilisation, mais dans des réseaux fixes. Ils ont illustré cette approche en proposant un algorithme d'exclusion mutuelle dont le temps de stabilisation est significativement meilleur lorsque l'exécution est synchrone. Nous adoptons ici une démarche similaire en montrant que, si l'on se restreint à $\mathcal{TC}^{\mathcal{B}}(\Delta)$ nos deux dernières solutions ont un temps de stabilisation asymptotiquement optimal, *i.e.*, $O(\Delta)$ rondes.

Par manque de place, nous ne pouvons détailler ici l'ensemble de nos résultats [†]. Dans la suite, nous avons choisi de présenter informellement les trois algorithmes que nous proposons.

2 Élections dans les TVG

Graphes dynamiques. Un TVG [3] est un quadruplet $\mathcal{G} = (V, E, \mathcal{T}, \rho)$ où V est un ensemble (statique) de nœuds, E est un ensemble (statique) d'arêtes entre paires de nœuds, $\mathcal{T}$ est un intervalle sur $\mathbb{N}^*$ nommé *durée de vie* de $\mathcal{G}$ et $\rho : E \times \mathcal{T} \mapsto \{0,1\}$ est la *fonction de présence* qui indique si une arête est présente à un instant donné. Dans un TVG, un *trajet* de p_0 à p_k est un chemin de p_0 à p_k au cours du temps, c'est-à-dire une suite $((p_0,p_1),t_0),((p_1,p_2),t_1),...,((p_{k-1},p_k),t_{k-1})$ telle que $\forall i \in \{0,...,k-1\}$, $\rho((p_i,p_{i+1}),t_i) = 1$ et $t_i < t_{i+1}$, pour $i < k-1$. La *distance temporelle* de p à q à un instant t est la durée minimale d'un trajet de p à q dont l'instant de départ est au plus tôt t. Par extension, le *diamètre temporel* est la distance temporelle maximale entre toutes paires de nœuds. Nous considérons ici les trois classes de TVG suivantes :

— Classe $\mathcal{TC}^{\mathcal{R}}$ (*connectivité temporelle*) : à tout moment, tout nœud peut atteindre les autres *via* un trajet (dont la durée n'est pas forcément bornée).

†. Un rapport technique complet est disponible sur HAL : `https://hal.archives-ouvertes.fr/hal-02376832/`.

— Classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$ (*diamètre temporel borné*) : à tout moment, tout nœud peut atteindre les autres *via* un trajet dont la durée est au plus Δ. Autrement dit, le diamètre temporel est borné par Δ.
— Classe $\mathcal{TC}^{\mathcal{Q}}(\Delta)$ (*diamètre temporel quasiment borné*) : infiniment souvent, un nœud peut atteindre les autres *via* un trajet dont la durée est au plus Δ.

Par définition, $\mathcal{TC}^{\mathcal{B}}(\Delta) \subseteq \mathcal{TC}^{\mathcal{Q}}(\Delta) \subseteq \mathcal{TC}^{\mathcal{R}}$.

Modèle de calcul. La topologie du réseau est représentée par un TVG dont les nœuds sont les processus et les arêtes sont les liens de communication (qui apparaissent et disparaissent au cours du temps). Les calculs s'effectuent par *rondes synchrones*. À chaque ronde, les processus communiquent en s'échangeant des messages mais sans connaître la topologie. Autrement dit, lorsqu'un processus p envoie un message au début de la ronde i, tous les processus qui sont ses voisins à la ronde i reçoivent ce message et le traitent pendant la ronde i.

L'élection. Nous proposons des algorithmes autostabilisants pour le problème de l'élection dans les classes de TVG définies ci-dessus. Pour cela, nous supposons que le *système distribué* est composé de n processus identifiés : l'*identité* de chaque processus p est noté $id(p)$. Le but de chaque algorithme est d'élire le processus d'identité minimum, noté ℓ. Ainsi, chaque processus p aura une variable $lid(p)$ permettant de stocker l'identité du processus qu'il considère comme étant l'élu.

Bien entendu, comme les solutions étudiées sont autostabilisantes, en début d'exécution, les variables commencent avec des valeurs arbitraires. Cela implique que $lid(p)$ peut contenir initialement une valeur qui n'est pas l'identité d'un processus du réseau ; nous appelons de telles valeurs des *fausses identités*. La stratégie mise en place par les algorithmes présentés va être dans un premier temps d'éliminer les fausses identités puis de calculer le minimum parmi les identités restantes.

Élection dans $\mathcal{TC}^{\mathcal{B}}(\Delta)$. Nous présentons d'abord un algorithme qui fonctionne dans la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$, où à tout moment, tout processus peut communiquer avec tout autre en au plus Δ rondes. Dans cet algorithme, chaque processus p connaît Δ (mais n'a pas besoin de connaître n). Chaque processus p est muni d'une variable $ttl(p) \in \{0,...,2\Delta\}$ qui permet de dater son leader actuel $lid(p)$. À chaque ronde, tout processus p envoie le message $\langle lid(p), ttl(p)\rangle$ et peut recevoir de ses voisins actuels des messages de la même forme. Dans ce cas, p met à jour sa paire de variables $(lid(p), ttl(p))$ avec la plus petite paire parmi les messages reçus et sa paire actuelle (par ordre lexicographique). Ensuite, il incrémente $ttl(p)$ puis réinitialise cette paire à $(id(p),0)$ si $lid(p) \geq id(p)$ ou si $ttl(p) \geq 2\Delta$ (la valeur de $lid(p)$ est trop ancienne, donc p suspecte qu'il s'agit d'une fausse identité).

Examinons maintenant le comportement de l'algorithme. Le mécanisme précédent (qui efface la valeur courante de $lid(p)$ quand sa date est trop ancienne) permet en au plus 2Δ rondes d'effacer toutes les fausses identités du système. À ce moment (et pour toujours), $lid(\ell)$ vaut $id(\ell)$ et $ttl(\ell)$ vaut 0. À partir de là, il faut au plus Δ rondes supplémentaires pour que la valeur $id(\ell)$ se propage de façon à ce que tous les processus p considèrent ℓ comme élu ; à noter aussi que $ttl(p)$ vaudra au plus Δ. Ainsi, en au plus 3Δ rondes, ℓ est élu et le reste pour toujours puisque les dates $ttl(p)$ de tout processus p ne dépasseront plus jamais Δ.

L'algorithme qui vient d'être présenté est une solution autostabilisante au problème de l'élection dans un réseau dynamique de la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$. Pourvu que chaque processus ait connaissance de Δ, l'algorithme converge et son temps de stabilisation est asymptotiquement optimal, c'est-à-dire de l'ordre de Δ rondes.

Élection dans $\mathcal{TC}^{\mathcal{Q}}(\Delta)$. Maintenant, regardons la classe $\mathcal{TC}^{\mathcal{Q}}(\Delta)$: le diamètre temporel est quasiment borné, au lieu d'être borné comme dans $\mathcal{TC}^{\mathcal{B}}(\Delta)$, ce qui signifie qu'à tout moment, un processus a la garantie de pouvoir communiquer un jour avec tout autre en au plus Δ rondes. Nos résultats sur la taille-ambiguïté (*cf.* Contributions) ont montré que résoudre le problème de l'élection avec une solution autostabilisante nécessite que les processus aient une connaissance exacte du nombre de processus n. Nous supposons aussi qu'ils connaissent Δ.

En sus de sa variable $lid(p)$, chaque processus p est équipé d'une file $members(p)$ d'au plus n paires d'identités datées (id,t). L'algorithme n'a besoin que d'*entrer* des éléments dans la file. Pour entrer une paire (id,t) dans $members(p)$, celle-ci est ajoutée en début de file. Si id était déjà présente dans $members(p)$ alors la paire (id,t') correspondant est retirée et la date associée à id rafraîchie pour être la plus petite des deux dates t et t' ; sinon si la file dépasse n éléments, le dernier élément est retiré. À chaque ronde, les processus envoient toutes les paires (id,t) de leur file telles que $t < \Delta$. Chaque paire reçue est entrée dans

la file en suivant la méthode ci-dessus ; toutes les dates de la file sont incrémentées ; puis la paire $(id(p),0)$ est entrée ; enfin $lid(p)$ est évalué comme le minimum des identités stockées dans la file.

Comme pour le premier algorithme, celui-ci purge dans un premier temps les fausses identités. En effet, les dates des fausses identités ne font que croître et après au plus Δ rondes, aucun processus ne relaie ni n'ajoute à sa file une fausse identité. Comme l'identité de chaque processus est ajoutée à sa propre file avec la date 0 à chaque ronde, toutes les vraies identités finissent par être insérées dans la file de chaque processus puisque la classe assure qu'un chemin temporel de longueur bornée par Δ va exister : ainsi, chaque file contiendra, en temps fini exactement l'ensemble des identités du réseau et la variable $lid(p)$ sera positionnée par tous les processus à $id(\ell)$.

Ainsi, cet algorithme est une solution autostabilisante au problème de l'élection dans un réseau dynamique de la classe $\mathcal{TC}^{\mathcal{Q}}(\Delta)$, pourvu que chaque processus ait connaissance de n et de Δ. Maintenant, si cet algorithme est déployé sur un TVG de la classe $\mathcal{TC}^{\mathcal{B}}(\Delta)$, nous avons la garantie que l'identité de tout processus va atteindre n'importe quel autre processus en au plus Δ rondes, ce qui assure un temps de stabilisation d'au plus 2Δ rondes dans ce cas. En ce sens, l'algorithme est spéculatif.

Élection dans $\mathcal{TC}^{\mathcal{R}}$. Nous adaptons l'algorithme précédent pour qu'il fonctionne pour la classe $\mathcal{TC}^{\mathcal{R}}$. Bien sûr, les processus connaissent de nouveau exactement n (sans quoi aucune solution n'est possible), mais ils n'ont besoin d'aucune autre entrée ; en particulier Δ n'est pas connu. Notre approche est analogue : chaque processus p est muni d'une file $members(p)$ d'identités datées ; à chaque ronde, les éléments de la file sont envoyés ; chaque paire reçue est *entrée* dans la file, les dates sont incrémentées, la paire $(id(p),0)$ est ajoutée et enfin $lid(p)$ est calculée comme minimum des identités présentes dans $members(p)$. La différence entre les deux algorithmes est la façon dont sont *entrés* les éléments dans la file. Ici, *entrer* la paire (id,t) dans $members(p)$ signifie : rafraîchir la date de id si cette identité est déjà dans la file, sinon ajouter la paire. Si cela fait déborder la file, alors un élément de date maximum est retiré au préalable.

Le fonctionnement et les arguments de correction de cette solution sont similaires à ceux de l'algorithme précédent (purge des fausses identités puis collecte de l'ensemble des identités des processus). L'argument spéculatif est lui un peu différent. Regardons cet algorithme s'exécuter sur un réseau de $\mathcal{TC}^{\mathcal{B}}(\Delta)$: après la première ronde, tous les processus envoient $(id(p),0)$ à chaque ronde. Donc tout processus q reçoit une paire $(id(p),t)$ avec $t \leq \Delta$ en au plus t rondes. Comme les fausses identités à ce moment ont forcément des dates plus grandes que t, $id(p)$ est inséré dans la file de q et n'en sera jamais effacé. Dans ces conditions, nous obtenons un temps de stabilisation d'au plus $\Delta+1$ rondes, au prix de dates (c'est-à-dire d'une mémoire locale) non bornée.

Conclusion. Au-delà de leurs garanties de fonctionnement et de performance, il faut remarquer que les trois algorithmes précédents ne sont pas silencieux : une fois le processus élu, des variables (les dates en l'occurrence) continuent à évoluer. Cet aspect est difficile à éviter car l'élection est un problème statique et le résultat d'impossibilité de [2] implique qu'il n'existe pas de solution silencieuse autostabilisante pour de nombreux problèmes statiques dans la classe des TVG toujours connectés temporellement, cette classe étant incluse dans les classes que nous avons étudiées.

Références

[1] K. Altisen, S. Devismes, A. Durand, and F. Petit. Gradual stabilization. *JPDC*, 123 :26–45, 2019.

[2] N. Braud-Santoni, S. Dubois, M. Kaaouachi, and F. Petit. The next 700 impossibility results in time-varying graphs. *IJNC*, 6(1) :27–41, 2016.

[3] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *Inter. J. of Parall., Emergent and Dist. Systems*, 27(5) :387–408, 2012.

[4] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago J. Theor. Comput. Sci.*, 1997, 1997.

[5] S. Dubois and R. Guerraoui. Introducing speculation in self-stabilization : an application to mutual exclusion. In *PODC*, pages 290–298, 2013.

[6] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. L. Wong. Zyzzyva : Speculative byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4) :7 :1–7 :39, 2009.