Pour battre à l'unisson, il faut que tous les chemins viennent de Rome

Karine Altisen, Alain Cournier, Geoffrey Defalque, Stéphane Devismes

Pour battre à l'unisson, il faut que tous les chemins viennent de Rome[†]

Karine Altisen[1], Alain Cournier[2], Geoffrey Defalque[2] et Stéphane Devismes[2]

[1] *Université Grenoble Alpes, VERIMAG (UMR 5104), Grenoble (France)*
[2] *Université de Picardie Jules Verne, MIS (UR 4290), Amiens (France)*

La littérature portant sur l'autostabilisation dans les réseaux dirigés est nettement moins fournie que celle sur les réseaux non-dirigés. C'est encore plus vrai lorsque l'on se restreint aux problèmes dynamiques où, à notre connaissance, seules les topologies en anneau unidirectionnel ont été considérées. Notre but ici est d'étudier les conditions sous lesquelles un algorithme peut résoudre de manière autostabilisante un problème dynamique simple : l'unisson synchrone.

Mots-clefs : Autostabilisation, réseaux dirigés, unisson synchrone.

1 Introduction

Contexte. Considérons un réseau anonyme, bidirectionnel et connexe. Supposons que dans ce réseau, les nœuds communiquent de manière synchrone via des variables localement partagées. Précisément, à chaque ronde, les nœuds exécutent simultanément les deux étapes suivantes : d'abord, ils lisent leur état et celui de leurs voisins dans le réseau ; puis ils modifient leur état en fonction de cette lecture. Arora *et al.* [3] ont proposé un algorithme très simple réalisant un *unisson synchrone* dans ce type de système. Dans ce problème, chaque nœud p dispose d'une horloge locale (une variable entière) $p.C$ dont le domaine de définition est $\{0, \ldots, K-1\}$, où $K > 1$ est une constante commune à tous les nœuds appelée *période*. Les nœuds doivent incrémenter régulièrement leurs horloges tout en les maintenant parfaitement synchronisées. L'algorithme local de chaque nœud p consiste, lors de chaque ronde, à affecter son horloge $p.C$ à $(h+1) \bmod K$ où h est le minimum entre sa valeur d'horloge et celles de ses voisins.

Bien que très simple, la règle exécutée par les nœuds semblent inutilement sophistiquée alors qu'*a priori* les nœuds vont initialiser leur horloge à zéro et qu'à partir d'une telle configuration initiale, une simple incrémentation modulo K à chaque ronde est suffisante pour résoudre le problème. En fait, cet algorithme réalise une propriété supplémentaire intéressante. Si les horloges sont initialisées avec des valeurs quelconques prises dans leur domaine de définition (*i.e.*, $\{0, \ldots, K-1\}$) et que K est supérieure ou égale à $\max(2, 2\mathcal{D}-1)$, où $\mathcal{D}$ est le diamètre du réseau, alors le système retrouvera en temps fini une configuration où toutes les horloges sont synchronisées. Autrement dit, si $K \geq \max(2, 2\mathcal{D}-1)$, alors l'algorithme d'Arora *et al.* est un algorithme *autostabilisant* d'unisson synchrone. En effet, à partir d'une configuration initiale quelconque, un algorithme autostabilisant garantit que le système atteindra en temps fini une configuration dite *légitime* à partir de laquelle son comportement sera conforme à sa spécification (ici, le fait que les horloges s'incrémentent régulièrement tout en restant synchronisées). 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*.

Dans cet article, nous proposons d'étudier le comportement de l'algorithme d'Arora *et al.* dans des réseaux dirigés, c'est-à-dire, des réseaux où l'existence d'un canal permettant la communication d'un nœud p vers un nœud q n'implique pas nécessairement l'existence d'un canal de communication dans le sens inverse. Ce type de réseau est une généralisation des réseaux bidirectionnels. En effet, dans un réseau dirigé, chaque lien de communication est soit bidirectionnel soit unidirectionnel. Lorsqu'un lien est orienté du nœud p vers

[†] Cet article est un résumé étendu de [2]. Il a été soutenu par le projet ANR SkyData (ANR-22-CE25-0008).

‡. Les fautes transitoires sont rares, de durée finie et affectent l'état du composant du réseau (nœud ou lien) où elles surviennent. La corruption de la mémoire d'un nœud ou du contenu d'un message sont deux exemples de fautes transitoires.

le nœud q, p (resp. q) est appelé *prédécesseur* (resp. *successeur*) de q (resp. p). Notez que si p et q sont reliés par un lien bidirectionnel alors ils sont prédécesseurs l'un de l'autre. Dans notre modèle à mémoires partagées, un nœud peut uniquement lire son état et celui de ses prédécesseurs. Ainsi, l'algorithme d'Arora *et al.* doit être légèrement adapté pour ce nouveau contexte : lors de chaque ronde, l'horloge de chaque nœud sera affectée à $(h+1) \bmod K$ où h est le minimum entre sa valeur d'horloge et celles de ses prédécesseurs. Cette version de l'algorithme d'Arora *et al.* sera nommée l'algorithme $\mathcal{U}$ dans la suite.

État de l'art. La motivation principale de ce travail est que l'autostabilisation dans les réseaux dirigés n'a pour le moment été que peu étudiée bien que ceux-ci soient très courants de nos jours (par exemple, les réseaux de capteurs sont généralement des réseaux dirigés). Précisément, la plupart des algorithmes autostabilisants pour des réseaux purement dirigés se sont concentrés sur des topologies en *anneau unidirectionnel* [5, 6] ou des *réseaux fortement connexes* [1, 4]. De plus, dans ce dernier cas, la littérature autostabilisante s'est uniquement intéressée à des *problèmes statiques*, c'est-à-dire, des problèmes calculant une propriété du réseau ou construisant une structure sur le réseau (comme un arbre couvrant, par exemple).

Contribution. Notre but ici est de trouver des topologies plus générales que les réseaux fortement connexes où l'autostabilisation d'un problème *dynamique* (c'est-à-dire, non-statique) peut être réalisée. Nous nous focalisons sur l'unisson car il s'agit d'un problème dynamique très simple qui, en outre, est une brique de base souvent utilisée dans la conception d'algorithme autostabilisant plus complexe.

Malheureusement, l'algorithme $\mathcal{U}$ ne stabilise pas dans des réseaux dirigés très simples, quelle que soit la période $K > 1$ choisie, comme illustré dans la figure 1 : à partir d'une configuration initiale bien choisie, les horloges ne se synchronisent pas durant les K premières rondes, or après celles-ci le système retrouve sa configuration initiale ; cela donne une exécution infinie où les horloges ne se synchronisent jamais.

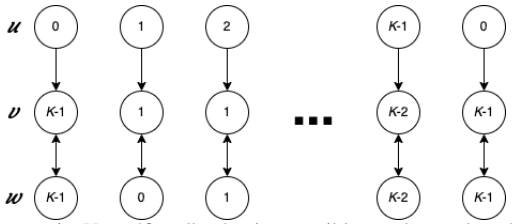

FIGURE 1 : Un préfixe d'exécution possible : u n'a pas de prédécesseurs, donc il incrémente son horloge modulo K à chaque ronde ; v a pour prédécesseurs u et w, par conséquent il affecte son horloge $v.C$ à $(\min\{u.C, v.C, w.C\} + 1) \bmod K$ à chaque ronde ; w a v pour unique prédécesseur, ainsi son horloge $w.C$ est assignée à $(\min\{v.C, w.C\} + 1) \bmod K$ lors de chaque ronde.

Nous avons alors étudié les conditions précises sous lesquelles l'algorithme $\mathcal{U}$ stabilise. Ces conditions dépendent de deux paramètres : la période et la topologie. Nous avons aussi démontré des bornes de complexité précises sur le temps de stabilisation de $\mathcal{U}$ valables dès lors que les conditions de stabilisation sont vérifiées. Cependant, par manque de place, nous allons présenter ici uniquement les conditions assurant la stabilisation de $\mathcal{U}$.

Plan. Dans la section suivante, nous présentons une condition nécessaire pour tout algorithme d'unisson synchrone autostabilisant. Dans la section 3, nous nous intéressons aux conditions précises garantissant la stabilisation de l'algorithme $\mathcal{U}$. Dans la dernière section, nous concluons avec quelques perspectives.

2 Condition nécessaire générale

Nous avons tout d'abord proposé une condition sur la topologie du réseau qui est nécessaire pour tout algorithme d'unisson synchrone autostabilisant. Cette condition utilise la notion de *graphe quotient*. Soit $G = (V, E)$ la topologie du réseau. Le graphe quotient de G est le graphe dont les nœuds représentent les composantes fortement connexe de G et dont chaque arc représente l'existence d'un arc de G reliant deux nœuds de composantes fortement connexes différentes. Notez que, par définition, tout graphe quotient est nécessairement sans circuit. Ainsi, on peut définir une notion de hauteur dans ce graphe : un nœud source (*i.e.*, un nœud sans prédécesseur) aura une hauteur de 0, la hauteur d'un nœud non-source sera égale à $h + 1$ où h est la hauteur maximale parmi ses prédécesseurs. Par abus de langage, on appellera dans la suite *hauteur* d'une composante fortement connexe, la hauteur du sommet associé à la composante dans son graphe quotient (par exemple, une composante fortement connexe associée à un nœud source dans son graphe quotient aura la hauteur 0).

On appelle *graphe à composante source unique* tout graphe dont le graphe quotient associé à un seul nœud source (un graphe à composante source unique et son graphe quotient associé sont donnés dans la figure 2).

Nous avons ensuite démontré que l'unisson synchrone admet une solution stabilisante seulement si la

topologie du réseau est un graphe à composante source unique. Cette condition est nécessaire même si le réseau est identifié et même si on utilise des horloges non-bornées (*i.e.*, sans le modulo K). Intuitivement, si le réseau contient au moins deux composantes sources C_1 et C_2, alors il est impossible de transmettre de l'information de C_1 à C_2 et inversement. Ainsi, elles agissent indépendamment l'une de l'autre et on ne peut pas garantir que ces deux composantes finiront par s'accorder sur une valeur d'horloge commune.

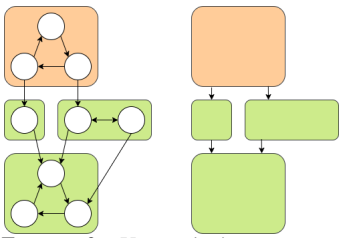

FIGURE 2 : Un graphe à composante source unique (à gauche) et son graphe quotient associé (à droite). En orange, nous représentons à gauche l'unique composante source et à droite le sommet source associé dans le graphe quotient. Les autres composantes fortement connexes sont en vert.

3 Conditions propres à l'algorithme

Malheureusement, la condition nécessaire précédente n'est pas suffisante pour assurer la stabilisation de $\mathcal{U}$: notamment, le contre-exemple de la figure 1 se déroule dans un réseau à composante source unique. Nous avons alors étudié sous quelles conditions la stabilisation de $\mathcal{U}$ est garantie. Notre résultat est le suivant : si $K \geq \max(2, 2\mathcal{D}^{\text{SCC}} - 1)$ où $\mathcal{D}^{\text{SCC}}$ est le diamètre maximum d'une composante fortement connexe de G, alors

— l'algorithme $\mathcal{U}$ stabilise si et seulement si G est un *graphe à composante source unique dense*.

Les graphes à composante source unique dense sont les graphes à composante source unique qui vérifient la condition supplémentaire suivante : tout sommet dans une composante non-source doit avoir au moins un prédécesseur dans une autre composante fortement connexe. Par exemple, le graphe de la figure 3 est à composante source unique dense. En revanche, celui de la figure 2 est à composante source unique mais pas dense. En effet, le nœud le plus à droite, par exemple, n'a pas de prédécesseur hors de sa composante fortement connexe.

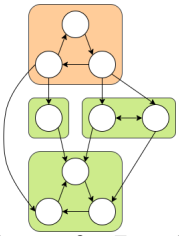

FIGURE 3 : Exemple de graphe à composante source unique dense.

La condition est suffisante. La preuve que cette condition est suffisante consiste tout d'abord à remarquer que l'exécution dans l'unique composante source du réseau est indépendante du reste du réseau car aucun nœud de cette composante n'a accès à l'état d'un nœud hors de la composante. Ensuite, le fonctionnement de l'algorithme $\mathcal{U}$ dans un réseau fortement connexe est très similaire à son comportement dans un réseau bidirectionnel. Ainsi, en adaptant légèrement les preuves de [3], on peut démontrer que l'ensemble des nœuds de la composante source finissent par se synchroniser grâce à la borne sur K choisie, qui est une généralisation de celle considérée dans [3].

Une fois synchronisées, la prochaine fois que les horloges des nœuds de la composante source atteindront la valeur 0 (après au plus K étapes), la synchronisation se propagera lors de la ronde suivante aux composantes fortement connexes de hauteur immédiatement supérieure (c'est-à-dire de hauteur 1) car tout nœud de ces composantes a au moins un prédécesseur dans la composante source. En poursuivant ce raisonnement par induction sur la hauteur des composantes fortement connexes dans le graphe quotient, on obtient une progression d'une hauteur supplémentaire toutes les K rondes et ainsi on démontre la stabilisation de l'ensemble du réseau.

Précision de la condition. Nous avons ensuite démontré la précision de notre condition. Tout d'abord, nous avons montré que pour toute valeur de K vérifiant la condition, dès lors que le réseau n'est pas un graphe à composante source unique dense, il existe au moins une exécution qui ne stabilise pas. Un contre-exemple dans un réseau à composante source unique mais non dense est donné dans la figure 4.

Réciproquement, nous avons démontré que, même en respectant le critère topologique, il existait des contre-exemples dès lors que le critère sur K était violé. Pour montrer cela, nous avons considéré des réseaux à composante source unique denses particuliers : des anneaux unidirectionnels, des chaînes bidirectionnelles et des graphes « cuillères » ; un graphes cuillère étant une chaîne bidirectionnelle reliée à un circuit de trois sommets (*cf.* figure 5). Dans le détail, nous avons montré que $\mathcal{U}$ n'est pas autostabilisant pour $K = 3$ dans les anneaux unidirectionnels de taille impaire d'au moins 7 nœuds. Ensuite, dans toute chaîne de diamètre $\mathcal{D} \geq 2$, nous avons des contre-exemples pour chaque valeur de K paire inférieure à la borne $\max(2, 2\mathcal{D}^{\text{SCC}} - 1)$. Enfin, dans tout graphe

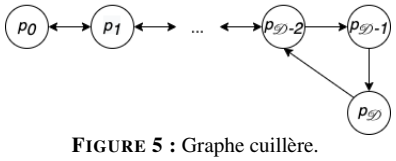

FIGURE 5 : Graphe cuillère.

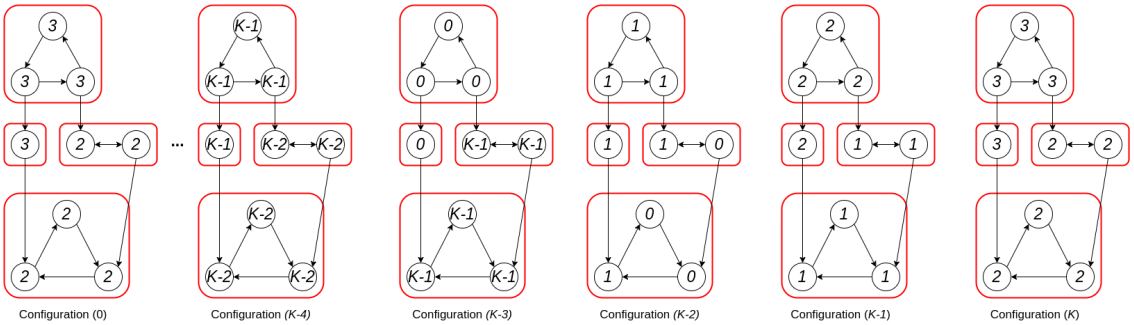

FIGURE 4 : Contre-exemple dans un réseau à composante source unique non dense.

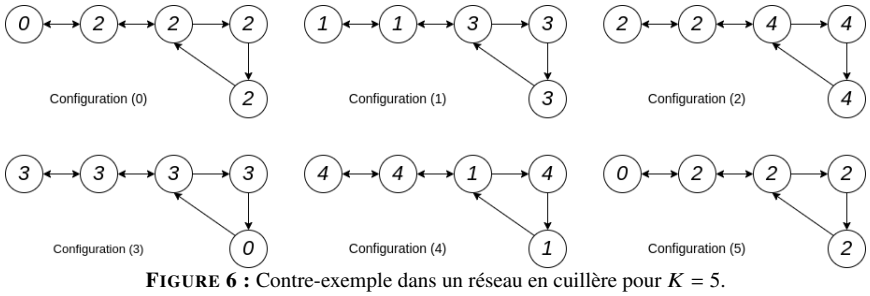

FIGURE 6 : Contre-exemple dans un réseau en cuillère pour $K = 5$.

cuillère de $\mathcal{D}+1$ nœuds avec $\mathcal{D} \geq 3$, c'est-à-dire, tout graphe cuillère de diamètre $\mathcal{D} \geq 3$, nous pouvons construire des contre-exemples pour chaque valeur de K impaire supérieure à 3 et inférieure à la borne $\max(2, 2\mathcal{D}^{\text{scc}} - 1)$; nous donnons un de ces contre-exemples dans la figure 6.

4 Conclusion

Nous avons défini les conditions précises garantissant la stabilisation de l'algorithme d'unisson synchrone d'Arora *et al.* dans les réseaux dirigés synchrones anonymes. Nous avons aussi démontré que son temps de stabilisation était en $O(H.K)$ rondes où H est la hauteur du graphe quotient du réseau. Enfin, nous avons exhibé une famille très importante de graphe à composante source unique dense où cette borne est atteignable.

Nos travaux montrent que l'autostabilisation est réalisable, en particulier pour les problèmes dynamiques, dans des classes de topologie plus générales que les graphes fortement connexes. L'ensemble de nos résultats a été démontré en raisonnant sur les graphes quotients, un outil indispensable pour étudier les réseaux dirigés non fortement connexes. Cette étude est, à notre connaissance, la première portant sur l'autostabilisation de problèmes dynamiques dans des classes de réseaux orientés si larges. Ainsi, nos futurs travaux tenterons d'étendre nos résultats à d'autres problèmes et sous des hypothèses plus faibles, notamment en considérant des modèles de communication asynchrones. Enfin, nous conjecturons que la condition nécessaire générale présentée ici est en fait suffisante lorsque les nœuds du réseau sont identifiés.

Références

[1] Y. Afek and A. Bremler-Barr. Self-stabilizing unidirectional network algorithms by power supply. *Chic. J. Theor. Comput. Sci.*, 1998, 1998.

[2] K. Altisen, A. Cournier, G. Defalque, and S. Devismes. Self-stabilizing synchronous unison in directed networks. In *ICDCN*, pages 115–124. ACM, 2023.

[3] A. Arora, S. Dolev, and M. G. Gouda. Maintaining digital clocks in step. *Parallel Process. Lett.*, 1 :11–18, 1991.

[4] S. Bernard, S. Devismes, M. Gradinariu Potop-Butucaru, and S. Tixeuil. Optimal deterministic self-stabilizing vertex coloring in unidirectional anonymous networks. In *IPDPS*, pages 1–8, Roma, Italia, 2009.

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

[6] H. Kakugawa and M. Yamashita. Uniform and self-stabilizing fair mutual exclusion on unidirectional rings under unfair distributed daemon. *JPDC*, 62(5) :885–898, 2002.