Auto-stabilisation dans les réseaux ad hoc †

N. Mitton[1], E. Fleury[1], I. Guérin-Lassous[1] et S. Tixeuil[2]

[1] *INRIA Ares - CITI, INSA de Lyon, 69621 Villeurbanne, France - prenom.nom@insa-lyon.fr*
[2]*LRI - CNRS UMR 8623 & INRIA Grand Large, 91405 Orsay, France - Sebastien.Tixeuil@lri.fr*

Dans cet article, nous choisissons un algorithme d'organisation de réseaux sans fil multi-saut, dit de *clusterisation*, et nous montrons que cet algorithme est auto-stabilisant. Nous proposons également certaines améliorations pour réduire le temps de stabilisation.

Keywords: réseaux sans fil multi-saut, clusterisation, auto-organisation, auto-stabilisation, extensibilité, DAG

1 Introduction

Les réseaux sans-fil multi-sauts sont des réseaux radio mobiles sans aucune infrastructure fixe. Leur fonctionnement est basé sur les principes d'auto-organisation et d'auto-stabilisation. Les protocoles actuels proposés pour un contexte multi-sauts fonctionnent correctement sur des réseaux de taille moyenne mais deviennent beaucoup trop coûteux sur de grands réseaux. Un défi majeur actuellement dans ce domaine concerne l'élaboration de solutions qui "passent à l'échelle", *i.e.* qui fonctionnent correctement sur de grands réseaux. Pour les protocoles de routage multi-sauts, une des solutions proposées pour tenter de résoudre ce problème de passage à l'échelle est d'organiser le réseau en regroupant géographiquement des nœuds proches en *clusters* (on parle alors de *clusterisation*) et en utilisant des approches différentes au sein des clusters et entre les clusters [KVCP97]. Dans la majorité des techniques de clusterisation proposées, chaque nœud choisit localement un chef. Tous les nœuds ayant choisi le même chef appartiennent au même cluster et le chef est appelé *cluster-head*. L'élection de ce chef peut se faire en se basant sur un critère d'identité (*p.ex.* le plus petit identifiant), sur un critère de connexité (*p.ex.* le degré ou le diamètre) ou sur un critère mixte identité/connexité. La maintenance de ces clusters repose elle aussi sur des critères fixes comme un diamètre ou rayon fixe ou un nombre constant de nœuds par cluster. Cependant, ces solutions ne sont pas adaptées aux grands réseaux puisque d'une part elles peuvent générer un nombre inutilement élevé de clusters et que d'autre part une toute petite modification de la topologie (comme le mouvement d'un nœud par exemple) peut conduire à de nouveaux calculs et reconstructions. Dans [MBF04], un nouveau critère de densité est proposé. Les auteurs montrent que cette métrique permet de limiter les reconstructions de clusters inutiles et se révèle plus stable que d'autres métriques face à la mobilité des nœuds.

Dans tous ces travaux de clusterisation, les auteurs testent la stabilité et la robustesse de leur algorithme face à la mobilité des nœuds par simulations, mais jamais selon une approche théorique. Dans cet article, nous cherchons à étudier la robustesse des algorithmes de clusterisation avec une approche théorique. Pour cela, nous partons de l'algorithme proposé dans [MBF04] puisqu'il présente de bonnes propriétés de robustesse et nous montrons que, sous certaines hypothèses, l'algorithme est auto-stabilisant. Nous proposons également l'ajout de certaines règles permettant d'améliorer la robustesse tout en conservant le caractère auto-stabilisant. Ces propriétés sont ensuite validées par simulations. Il faut noter que l'algorithme de [MBF04] est un point de départ et que cette étude pourrait aussi être appliquée à d'autres protocoles de clusterisation.

†Ces travaux sont soutenus par le FSN du ministère de la recherche via le projet FRAGILE de l'ACI Sécurité et Informatique.

2 Quelques rappels

Étant donnée la limitation sur le nombre de pages, il n'est pas possible de décrire notre étude dans son intégralité. Le lecteur intéressé par ce sujet peut se reporter à [MFGLT05]. Toutes les preuves des théorèmes et des lemmes suivants y sont décrites, ainsi que toutes les simulations réalisées. Dans cette section, nous donnons les notations utilisées par la suite, nous décrivons brièvement la métrique de densité choisie et la formation des clusters basée sur cette métrique et nous rappelons quelques pré-requis pour l'étude d'auto-stabilisation.

Modèle et hypothèses. Le système est modélisé par un ensemble de nœuds V. Chaque nœud p a un identifiant unique et peut communiquer avec un sous-ensemble $N_p \subseteq V$ de nœuds défini par la portée du signal radio de p ; N_p est appelé le *voisinage* du nœud p, $p \notin N_p$. Nous supposons les liens de communication bidirectionnels : $q \in N_p$ ssi $p \in N_q$. Définissons $N_p^1 = N_p$ et, pour $i > 1$, le i-voisinage de p $N_p^i = N_p^{i-1} \cup \{ r \mid (\exists q \in N_p^{i-1}, r \in N_q) \}$. De plus, nous supposons que la répartition des nœuds est telle qu'il existe une constante connue δ telle que pour tout nœud p, $|N_p| \leq \delta$. Ce contrôle de densité peut être réalisé en ajustant la portée de transmission des nœuds dans des zones trop peuplées.

La métrique de densité et formation des clusters. La notion de densité, introduite dans [MBF04] tente de caractériser l'importance « relative » d'un nœud dans son propre voisinage. Cette notion permet d'absorber de petits changements locaux de topologie, en considérant le rapport entre le nombre de liens et le nombre de nœuds dans un voisinage. La densité d'un nœud $p \in V$ est la suivante : $d_p = \frac{|e=(v,w) \in E \ tq. \ w \in \{p\} \cup N_p \ et \ v \in N_p|}{|N_p|}$. Chaque nœud calcule localement sa valeur de densité et la diffuse périodiquement à ses voisins. S'il possède une densité plus forte que tous ses voisins, il s'élit chef, sinon, il élit comme père son voisin de plus forte densité. En cas d'égalité, le plus petit identifiant départage les ex æquo. Le chef sera le nœud s'étant choisi comme son propre père.

Auto-stabilisation Une description plus complète de l'étude de l'auto-stabilisation d'un algorithme est donnée dans [HT04]. Nous gardons les mêmes hypothèses que dans [HT04], à savoir que nous disposons d'une implantation de CSMA/CA, pour l'accès au médium radio, qui satisfait les points suivants : il existe une constante $\tau > 0$ telle que la probabilité d'une transmission de trame sans collision est supérieure ou égale à τ. Nous décrivons les algorithmes sous la forme de règles gardées : $G \rightarrow S$ représente une telle règle, où G est un prédicat sur les variables locales d'un nœud, S une affectation de ces mêmes variables locales. Si le prédicat G (la garde) est vrai, l'affectation S est exécutée, sinon elle est ignorée. L'exécution du système consiste pour chaque nœud à évaluer répétitivement ses règles gardées. Nous supposons que chaque règle activable est exécutée en un temps constant. Certaines variables des nœuds sont dites *partagées*. Suivant le schéma présenté en [HT04], les nœuds diffusent périodiquement les valeurs de leurs variables partagées. Nous supposons également que le schéma de [HT04] est utilisé pour obtenir sur chaque nœud N_p et N_p^2.

3 Étude d'auto-stabilisation

3.1 Construction d'un DAG à hauteur constante

Dans l'algorithme choisi, comme dans tout algorithme utilisant l'identifiant des nœuds comme critère de décision finale, le pire cas se rencontre quand tous les nœuds ont la même valeur de décision (comme le degré ou la densité) et que les identifiants des nœuds sont uniques et mal distribués. L'algorithme peut alors ne construire qu'un seul cluster dont le diamètre est aussi grand que celui du réseau, le temps de stabilisation dépendant de ce diamètre. Pour pallier cet inconvénient, il peut s'avérer utile de donner aux nœuds d'autres identifiants, choisis dans un espace de noms constant et plus petit, de façon à ce que les identifiants soient localement uniques. Un DAG (Directed Acyclic Graph) peut alors être construit à partir de ces nouveaux noms en orientant les arêtes entre les voisins du plus grand identifiant vers le plus petit. Notre construction de DAG est basée sur la technique aléatoire décrite dans [HT04], mais utilise un espace

de noms beaucoup plus petit γ ($|\gamma| = \delta^6$ dans [HT04], tandis que δ^2 ou même δ est suffisant dans notre cas). Soit Id_p une variable partagée appartenant au domaine γ, correspondant au *nom* du nœud p dans le DAG. Soit $Cids_p = \{\bowtie Id_q \mid q \in N_p\}$, une variable pour déterminer le nom des nœuds voisins, où $\bowtie Id_q$ réfère à la copie en cache de la variable partagée Id_q sur le nœud p. Supposons que $\mathsf{random}(S)$ choisit avec une probabilité uniforme un élément de S. Le nœud p utilise la fonction suivante pour calculer Id_p :

$$\mathsf{newId}(Id_p) = \begin{cases} \bowtie Id_p & \text{si } \bowtie Id_p \notin Cids_p \\ \mathsf{random}(\gamma \setminus Cids_p) & \text{sinon} \end{cases}$$

L'algorithme de construction d'un DAG à hauteur constante est le suivant : $\mathsf{N1} : \mathit{true} \rightarrow Id_p := \mathsf{newId}(Id_p)$.

Theorème 1. *L'algorithme N1 stabilise avec probabilité* 1 *en un temps constant vers un DAG de hauteur inférieure ou égale à* $|\gamma| + 1$.

Un compromis est à faire pour déterminer le paramètre γ : plus la valeur de $|\gamma|$ est grande, plus le temps de convergence de N1 est faible mais plus la hauteur du DAG est importante. Une hauteur de DAG importante augmente le temps de stabilisation des règles R1 et R2 (*cf.* Section 3.2).

3.2 Formation de clusters suivant la densité

Chaque nœud p maintient deux variables partagées : d_p et $\mathcal{H}(p)$ où d_p est la densité du nœud p et $\mathcal{H}(p)$ son cluster-head. Nous définissons $\prec$ tel que $p \prec q$ si et seulement si $d_p < d_q$ ou $(d_p = d_q) \wedge (Id_q < Id_p)$. $\max_\prec$ représente la valeur du maximum de $\prec$. Quand un nœud p calcule $\prec$ ou $\max_\prec$, il utilise les valeurs de cache de son voisinage ($\bowtie Id_p = Id_p$ et $\bowtie d_p = d_p$).

Nous définissons maintenant la fonction d'élection du cluster-head :

$$\mathsf{clusterHead} = \begin{cases} Id_p & \text{if } \forall q \in N_p, q \prec p \\ \mathcal{H}(\max_\prec \{q \in N_p\}) & \text{sinon} \end{cases}$$

L'algorithme s'exécute comme suit :

$$\begin{array}{ll} \mathsf{R1}: & \mathit{true} \rightarrow d_p := \mathsf{density} \\ \mathsf{R2}: & \mathit{true} \rightarrow \mathcal{H}(p) := \mathsf{clusterHead} \end{array}$$

Lemme 1. *Partant de n'importe quelle configuration initiale, chaque nœud p a une valeur de densité* correcte *d_p en un temps borné constant.*

Lemme 2. *Partant de n'importe quelle configuration initiale, chaque nœud p a une valeur* correcte *pour $\mathcal{H}(p)$ en un temps borné constant.*

Remarque : Nous sommes également en train d'établir une borne supérieure sur le temps de convergence de l'algorithme.

3.3 Améliorer la stabilité

Afin d'améliorer la stabilité de notre algorithme, nous introduisons des règles de décision supplémentaires lors de l'élection du cluster-head. Premièrement, lorsque deux nœuds u et v sont en compétition pour devenir le père d'un troisième nœud w, l'élu sera prioritairement celui choisi précédemment par w (s'il existe), et sinon celui ayant le plus faible identifiant de DAG (comme défini dans la Section 3). Cette nouvelle règle préserve la structure de notre preuve de stabilisation, puisqu'il suffit de définir $\prec$ comme $p \prec q$ si et seulement si $(d_p < d_q)$ ou $(d_p = d_q) \wedge (\mathcal{H}(q) = Id_q) \wedge (\mathcal{H}(p) \neq Id_p)$ ou $(d_p = d_q) \wedge (\mathcal{H}(p) \neq Id_p) \wedge (\mathcal{H}(q) \neq Id_q) \wedge (Id_q < Id_p)$. De plus, la hauteur du nouveau DAG$_\prec$ est similaire à celle de l'ancien. Deuxièmement, si un nœud p est 1-voisin de deux cluster-heads différents u et v (qui ne sont donc pas voisins), il initie une fusion entre les clusters de u et v : si p a choisi v comme cluster-head, cela signifie que $u \prec v$ et donc v reste cluster-head contrairement à u. Cela assure que *(i)* un cluster-head n'est pas trop excentré dans son propre cluster, *(ii)* un cluster a un diamètre supérieur ou égal à 2, et *(iii)* que deux cluster-heads sont distants d'au moins trois sauts. Encore une fois, ces règles préservent notre preuve de stabilisation puisqu'il suffit d'utiliser la fonction alternative :

$$\mathsf{clusterHead} = \begin{cases} Id_p & \text{si } (\forall q \in N_p, q \prec p) \wedge (\forall q \in N_p^2 | \mathcal{H}(q) = Id_q \implies q \prec p) \\ \mathcal{H}(\max_\prec \{q \in N_p\}) & \text{sinon} \end{cases}$$

4 Simulations

D'après l'hypothèse réalisée sur l'accès au médium radio, nous pouvons dire que chaque nœud est en mesure de diffuser localement une trame et d'en recevoir une de chacun de ses voisins en un temps borné $\Delta(\tau)$, appelé une *étape*. Après une étape, chaque nœud connaît ses 1-voisins. Après deux étapes, il connaît ses 2-voisins et peut calculer sa valeur de densité et après trois étapes, il connaît son père. Le nombre d'étapes nécessaires à un nœud pour connaître l'identité de son cluster-head dépend directement de la distance qui l'en sépare et est borné par la profondeur de l'arbre. Pour la construction du DAG et l'attribution des *noms* des nœuds, chaque nœud se choisit aléatoirement un *nom* entre 0 et δ^2. Il compare alors ce nom à celui de ses voisins. Si deux voisins ont le même nom, le nœud dont l'identifiant "normal" est le plus petit se choisit un autre nom et ainsi de suite jusqu'à ce qu'il n'existe aucune paire de nœuds voisins portant le même nom.

Afin d'évaluer les performances de l'algorithme et d'estimer l'apport de l'introduction d'un DAG, nous avons réalisé des simulations où les nœuds sont soit déployés aléatoirement suivant un processus de Poisson dans un carré 1×1 avec plusieurs valeurs de portée de transmission R et d'intensité du processus λ soit disposés sur une grille. Chaque statistique est la moyenne sur 1000 simulations. Nous pouvons d'abord noter que construire un DAG n'est pas coûteux, puisque cela prend en moyenne deux étapes. Nous avons ensuite évalué le nombre de cluster-heads par unité de surface, la hauteur de l'arbre de clusterisation et l'excentricité du cluster-head dans son cluster en nombre de sauts. Pour le déploiement aléatoire, l'excentricité moyenne d'un cluster-head et la hauteur de l'arbre varient peu en fonction de la portée radio, donc un nœud connaît l'identité de son cluster-head en un temps faible et constant. Dans ce cas, l'utilisation d'un DAG n'est d'aucune aide car les valeurs de densité étant uniformément réparties et rarement égales, l'identifiant du nœud est rarement utilisé pour sélectionner le père. Dans le cas de la distribution sur une grille, les identifiants sont choisis croissant de la gauche vers la droite et du bas vers le haut. Tous les nœuds intérieurs ont alors la même densité et le choix du père se fait en fonction de l'identifiant. Dans un pareil cas, la construction du DAG est utile (cf Tab 1) car elle permet de réduire drastiquement le nombre d'étapes nécessaires avant la stabilisation (puisqu'elle réduit la hauteur des arbres de clusterisation).

	$R = 0.05$		$R = 0.08$		$R = 0.1$	
	Avec DAG	Sans DAG	Avec DAG	Sans DAG	Avec DAG	Sans DAG
# clusters	52.8	1.0	29.3	1.0	18.5	1.0
excentricité	3.4	29.1	4.1	19.1	3.6	6.5
hauteur d'arbre	3.7	83.4	4.7	100.5	4.5	32.1

TAB. 1: Caractéristiques des clusters dans une grille.

Pour tester les règles introduites en Section 3.3, nous avons réalisé des simulations où les nœuds bougent aléatoirement à des vitesses aléatoires. Nous avons calculé le pourcentage de cluster-heads restant cluster-heads après chaque série de mouvement. Pour une mobilité de nœud entre 0 et $1.6m/s$ (piétons), ce pourcentage est d'environ 82% avec nos règles supplémentaires contre 78% sans. Pour une mobilité de nœud entre 0 et $10m/s$ (voitures), ce pourcentage est de 31% avec les nouvelles règles contre 25% sans.

Références

[HT04] T. Herman and S. Tixeuil. A distributed tdma slot assignment for wireless sensor networks. In *Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, number 3121 in Lecture Notes in Computer Science, pages 45–58, Turku, Finland, July 2004. Springer-Verlag.

[KVCP97] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster based approach for routing in dynamic networks. In *ACM SIGCOMM*, pages 49–65, April 1997.

[MBF04] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *3nd Med-Hoc-Net*, june 2004.

[MFGLT05] Nathalie Mitton, Eric Fleury, Isabelle Guerin-Lassous, and Sebastien Tixeuil. Self-stabilization in self-organized multihops wireless networks. In *WWAN'05*, june 2005.