

Clustering pour réseaux spontanés basé sur le degré de connectivité

Fehmi Ben Abdesslem¹, Artur Ziviani³, Marcelo Dias de Amorim^{1,2} et Petia Todorova⁴

¹UPMC Univ Paris 06, UMR 7606, Laboratoire d'Informatique de Paris 6 (LIP6), F-75016 Paris, France

²CNRS, UMR 7606, Laboratoire d'Informatique de Paris 6 (LIP6), F-75016 Paris, France

³Laboratório Nacional de Computação Científica (LNCC), Av. Getúlio Vargas 333, 25651-075 Petrópolis, RJ, Brésil

⁴Fraunhofer Institut für Offene Kommunikationssysteme FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Allemagne

Dans les réseaux sans fil spontanés, la formation de clusters (grappes de nœuds) est souvent considérée comme une solution primordiale à l'amélioration des performances de plusieurs classes de protocoles de communication. La principale difficulté rencontrée par la plupart des algorithmes distribués existants est d'équilibrer la taille des clusters formés tout en garantissant une taille moyenne de ces clusters proche d'une valeur prédéfinie. Afin de parvenir à cet objectif, nous proposons dans cet article de former les clusters de façon distribuée en répartissant un nombre fixe de jetons (budget). La propagation de ces jetons prend en compte le degré de connectivité de chacun des nœuds, afin de créer des clusters en perdant un minimum de jetons. Nous montrons que les clusters ainsi formés présentent une taille plus proche du budget initial qu'avec les propositions antérieures.

Keywords: Clustering, réseaux spontanés sans fil

1 Introduction

Parmi les multiples mécanismes existants permettant aux algorithmes pour réseaux sans fil spontanés de fonctionner à grande échelle, le clustering fait partie de ceux ayant focalisé le plus d'attention de la part de la communauté de chercheurs [YKR06]. La plupart des algorithmes de formation de clusters se basent sur l'élection de *cluster-heads*. Cette élection, qui s'opère selon des critères tels que le degré de connectivité [KMW04] ou la quantité d'énergie restante [YF04] conduit souvent à la formation de clusters de taille très variable et difficile à contrôler. Dans cet article, nous présentons un nouvel algorithme de clustering qui consiste à former de manière distribuée des groupes de nœuds à partir d'une topologie aléatoire, en se basant sur un budget initial à répartir sous forme de jetons. Ainsi, l'avantage de cette technique réside dans le contrôle de la taille des clusters formés.

Comme d'autres algorithmes de formation de clusters basés sur des budgets [KS06], notre algorithme commence par l'affectation de β jetons (budget) à un ensemble de nœuds initiateurs, aléatoirement choisis, qui feront également office de *cluster-heads*. Le budget β correspond à la taille idéale des clusters. Quant au choix des nœuds initiateurs, il se fait de manière distribuée en armant une minuterie sur chaque nœud, réglée avec un temps aléatoire. A l'expiration de sa minuterie, un nœud déclenche la formation du cluster en distribuant le budget initial à ses voisins directs. Lorsqu'un nœud reçoit un jeton avant l'expiration de sa minuterie, celle-ci est aussitôt désarmée. Les voisins vont à leur tour recruter de nouveaux nœuds pour former le cluster, en redistribuant à leurs voisins respectifs les jetons qui leur ont été attribués, et ceci récursivement jusqu'à épuisement du budget. La convergence de l'algorithme est garantie par la consommation d'un jeton par chacun des nœuds recrutés. Dans le meilleur des cas, chaque cluster ainsi formé a une taille égale à β .

Former des clusters dont la taille est la plus proche possible du budget de départ est une problématique clé. Des algorithmes de clustering existants basés sur des budgets tels que Rapid ou Persistent [KS06], distribuent aveuglément le budget au voisinage, à parts égales. Ainsi, un nœud n'ayant aucun autre voisin à recruter peut se voir allouer inutilement des jetons à redistribuer. Ces jetons perdus auraient pu être utilisés par un autre nœud avec un fort degré de connectivité mais n'ayant pas reçu suffisamment de jetons pour

étendre le cluster à ses voisins. Ce cas de figure apparaît souvent dans des topologies dans lesquelles la densité varie à travers le réseau.

Nous proposons un nouvel algorithme de clustering basé sur des budgets, qui prend en compte les degrés de connectivité de chaque nœud avoisinant pour pondérer la distribution des jetons. Cet algorithme, que nous appelons PBC (*Potential-Based Clustering*), produit des clusters dont la taille est bien plus proche du budget β par rapport aux approches existantes.

2 Clustering basé sur la connectivité (PBC)

Dans les algorithmes de clustering basés sur des budgets, un nœud initiateur dispose au préalable d'un budget alloué β , qu'il commence par consommer en s'attribuant une unité, avant de distribuer les $\beta - 1$ restants à ses voisins. Ses voisins consomment à leur tour une unité du budget reçu, et redistribuent les jetons restants à leurs voisins (en dehors de la source). Les redistributions sont ensuite effectuées récursivement jusqu'à épuisement du budget ou absence de voisins disponibles.

La distribution récursive du budget à parts égales provoque inévitablement une affectation excessive de jetons à certains voisins, entraînant la perte de jetons et la réduction de la taille du cluster en formation. Se limiter à un simple retour de ces jetons superflus, comme dans l'algorithme Persistant de Krishnan et al., permet d'augmenter sensiblement la taille moyenne des clusters, mais provoque la formation de clusters étendus et enchevêtrés les uns aux autres. Ce phénomène s'explique par une mauvaise distribution des jetons, qui entraîne l'isolement de nœuds en marge des clusters. Ces nœuds isolés, dont la plupart des voisins ont déjà été recrutés par la formation d'un précédent cluster, ne trouveront pas assez de voisins disponibles pour écouler l'ensemble du budget. Afin d'éviter la perte de budget et la création de clusters de taille réduite, nous proposons, en plus de retourner les jetons superflus, de ne plus les distribuer aveuglément.

Nous confondons par la suite le degré de connectivité d'un voisin avec son potentiel de distribution. Ainsi, nous nous appuyons sur le principe selon lequel un nœud voisin présenterait un potentiel de distribution proportionnel à son degré de connectivité. Pour ce faire, nous partons donc de l'hypothèse que si un nœud A a plus de voisins qu'un nœud B , alors le sous-arbre associé à A est probablement plus grand (en nombre de nœuds) que le sous-arbre associé à B . Dans la Partie 3, nous montrons expérimentalement à l'aide de simulations que cette hypothèse est vérifiée assez souvent pour permettre une amélioration des performances sur des topologies aléatoires uniformément distribuées.

Un pré-requis à l'algorithme est donc la connaissance par chaque nœud du degré de connectivité de ses voisins. Celui-ci peut être facilement déterminé en utilisant les balises envoyées régulièrement par les nœuds. Soit $\mathbf{N}(X) = \{x_1, x_2, \dots\}$ l'ensemble des voisins de X . La taille de cet ensemble définit le potentiel du nœud X , noté $\pi(X) = |\mathbf{N}(X)|$. Soit A un nœud disposant d'un budget β_A . La quantité de jetons alloués à son voisin $a_i \in \mathbf{N}(A)$ est calculée de la manière suivante :

$$\beta_{a_i} = \left\lfloor \frac{(\beta_A - 1)\pi(a_i)}{\sum_{j=1}^{|\mathbf{N}(A)|} \pi(a_j)} \right\rfloor. \quad (1)$$

Le nœud X commence d'abord par consommer un jeton et distribue le reste en se basant sur les potentiels de ses voisins. Un voisin bénéficiant d'un fort potentiel recevra donc plus de jetons qu'un voisin de plus faible potentiel. Comme évoqué précédemment, sur des topologies aléatoires, il arrive fréquemment que des nœuds n'appartenant pas encore à un cluster voient la totalité de leurs voisins recrutés par d'autres clusters. A l'expiration de sa minuterie, un tel nœud est alors contraint de former son propre cluster, sans pouvoir recruter de voisins. Ce cas de figure nuit lourdement aux performances de l'algorithme, puisqu'il diminue la taille moyenne des clusters en en formant un composé d'un seul nœud.

Pour cette raison, notre algorithme a été conçu pour former des clusters plus compacts, afin d'éviter l'apparition de clusters de petite taille. Ainsi, d'après l'équation 1, notons que certains jetons peuvent ne pas être attribués, si $(\beta_A - 1)$ n'est pas multiple de $\sum_{j=1}^{|\mathbf{N}(A)|} \pi(a_j)$. Dans ce cas, les jetons restants sont répartis uniformément, en commençant par les voisins de faible potentiel. Expérimentalement, on observe que la propagation des jetons évite les nœuds à faible potentiel, provoquant ainsi la création de clusters de taille réduite dans les zones de faible densité. Choisir de privilégier ici les voisins de faible potentiel pour distribuer les jetons restants permet d'obtenir de meilleurs résultats en accordant à ceux-ci une chance de recevoir quelques

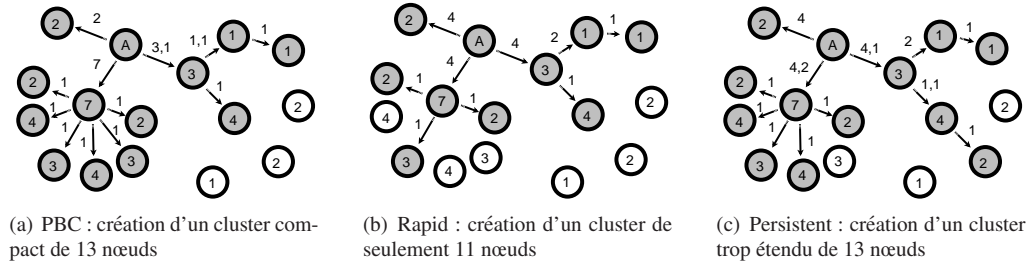


FIG. 1: Comparaison illustrative des différents algorithmes. L'inondation est initiée par le nœud A. Les liens à plusieurs valeurs sont sujets à de multiples distributions consécutives et l'étiquette de chaque nœud correspond à son potentiel.

jetons complémentaires afin de rejoindre un cluster voisin. En pratique, soit $\beta_r = (\beta_A - 1) - \sum_{j=1}^{|\mathcal{N}(A)|} \beta_{a_j}$. Pour les β_r voisins de plus faible potentiel, le nœud A effectue l'addition $\beta_{a_i} \leftarrow \beta_{a_i} + 1$ avant de distribuer le budget. Pour des raisons similaires, recruter tous les voisins à chaque distribution est essentiel pour éviter que l'un d'eux ne forme plus tard son propre cluster (à l'expiration de sa minuterie) sans trouver de voisins disponibles. Ainsi, avant chaque distribution, l'algorithme commence par attribuer un jeton à chacun des voisins. Les éventuels jetons restants sont ensuite répartis selon l'équation 1. S'il reste des jetons non attribués, alors ceux-ci sont attribués en priorité aux nœuds à faible potentiel, comme décrit précédemment.

Malgré ces mesures, surtout efficaces pour des budgets importants, certains clusters sont encore composés d'un seul nœud lorsque le budget est plus réduit. Pour y remédier, notre algorithme prévoit le rattachement à un cluster voisin des nœuds sans aucun voisin disponible. Ainsi, lorsque le dernier voisin disponible d'un nœud est recruté par un cluster, ce nœud s'associe également au cluster en formation. Nous introduisons de cette manière une certaine flexibilité au budget initial, qui n'est plus une borne maximale à la taille des clusters mais plutôt la valeur cible pour la taille moyenne des clusters formés.

Afin de donner un aperçu du fonctionnement de notre algorithme, nous le comparons avec deux autres algorithmes basés sur les budgets trouvés dans la littérature, à savoir Rapid et Persistent [KS06]. Dans l'algorithme Rapid, les jetons non attribués ne sont pas rendus au *cluster-head*. Cet algorithme bénéficie ainsi d'une complexité minimale en ce qui concerne le nombre de messages émis, au prix cependant d'une efficacité limitée. Dans l'algorithme Persistent, l'idée centrale réside dans le retour des jetons superflus afin qu'il soient redistribués ailleurs.

Nous montrons dans la figure 1 comment PBC se comporte comparé à Rapid et Persistent, avec un budget $\beta = 13$. La figure 1(b) représente l'exécution de l'algorithme Rapid, avec lequel seuls 10 nœuds composent le cluster de A. La figure 1(c) illustre l'exécution de l'algorithme Persistent. La seconde valeur affectée à certains liens représente une seconde distribution inhérente au retour de jetons superflus. Enfin, la figure 1(a) représente l'exécution de notre algorithme. Pour plus de clarté, seule la distribution issue de la formule 1 a été utilisée, sans les mesures complémentaires destinées à éviter que des nœuds adjacents ne créent des clusters de petite taille. La figure montre que notre algorithme obtient le même résultat maximal que Persistent, mais s'efforce d'attribuer des jetons proportionnellement au degré de connectivité de chaque voisin afin de former des clusters plus compacts. Ainsi, les 3 nœuds encore disponibles sont moins dispersés dans la topologie, et peuvent donc appartenir à un même cluster. En revanche, Persistent sera plus souvent confronté à des nœuds isolés, qui devront chacun de leur côté créer un nouveau cluster en initiant une autre inondation, diminuant ainsi la taille moyenne des clusters.

3 Simulations

Pour comparer les algorithmes, des simulations ont été effectuées sur des topologies générées aléatoirement. Le simulateur utilisé est NS-2, et les topologies aléatoires sont générées sur une surface circulaire de 2500m de rayon, chaque nœud ayant une couverture radio de 100m. La densité moyenne de la topologie varie de 2 à 8 nœuds par zone de couverture, soit de 1250 à 5000 nœuds. Les topologies ont été générées aléatoirement selon une distribution uniforme, à l'aide de Topogen [HCG08]. Pour chaque densité, une nouvelle topologie aléatoire est générée. Les trois algorithmes sont ensuite exécutés sur cette même topologie.

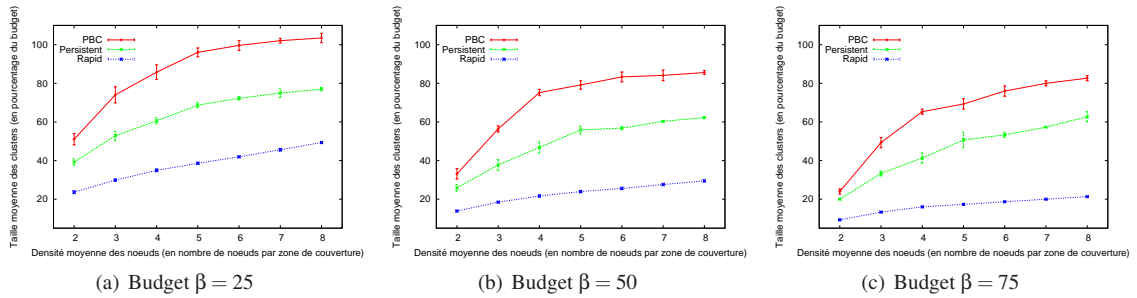


FIG. 2: Taille moyenne des clusters en fonction de la densité des topologies aléatoires.

L'efficacité d'un algorithme de clustering basé sur les budgets s'évalue en fonction de la taille des clusters formés. Le cas idéal serait d'obtenir des clusters de taille égale au budget. Cependant, la propagation des jetons ne permettant pas toujours d'atteindre une telle taille pour tous les clusters, la taille moyenne des clusters reste en général bien inférieure au budget. Ainsi, plus cette taille moyenne est proche du budget, plus l'algorithme est efficace. Les résultats des simulations montrent que la taille moyenne des clusters est jusqu'à 20% plus importante lorsque les jetons sont distribués en tenant compte des potentiels, pour une densité de 8 nœuds par zone de couverture, et un budget de 75. Cependant, pour la même densité, l'amélioration est moins nette pour des budgets plus faibles : 9% pour un budget de 50, et aucune amélioration significative pour un budget de 25. Pour obtenir des résultats quel que soit le budget, notre algorithme comprend également des mécanismes simples, décrits dans la partie précédente (e.g. flexibilité du budget). La figure 2 représente les résultats de notre algorithme avec l'ensemble de ses mécanismes, et pour un budget fixé à $\beta = 25, 50$ et 75 . Chaque valeur correspond à une moyenne sur 5 simulations. L'algorithme Rapid est le moins efficace puisqu'il distribue les jetons aveuglément sans mécanisme de retour. En renvoyant les jetons non alloués, Persistent parvient à de meilleurs résultats. Cependant, la taille moyenne des clusters est de 33.8 % plus importante en utilisant notre algorithme PBC (pour un budget de 75 et une densité de 8), mais également 35.48 % et 33.6 % pour des budgets plus faibles de 50 et 25, respectivement.

4 Conclusion

Les algorithmes de clustering basés sur des budgets peuvent être optimisés en prenant en compte les degrés de connectivité des voisins lors de la distribution des jetons. Pour cela, nous proposons l'algorithme PBC (*Potential-Based Clustering*), qui comprend également d'autres mécanismes permettant d'obtenir des clusters de taille plus proches du budget. Des simulations confirment l'efficacité de cette approche sur des topologies uniformément aléatoires. Les travaux en cours visent à définir de manière distribuée le meilleur sous-ensemble de nœuds initiateurs optimisant les performances.

Références

- [HCG08] E. Ben Hamida, G. Chelius, and J.-M. Gorce. Scalability versus accuracy in physical layer modeling for wireless network simulations. In *22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS)*, Rome, Italy, June 2008.
- [KMW04] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *MOBICOM*, pages 260–274, Philadelphia, USA, 2004.
- [KS06] Rajesh Krishnan and David Starobinski. Efficient clustering algorithms for self-organizing wireless sensor networks. *Ad Hoc Networks*, 4(1) :36–59, 2006.
- [YF04] Ossama Younis and Sonia Fahmy. Heed : A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Trans. Mob. Comput.*, 3(4) :366–379, 2004.
- [YKR06] Ossama Younis, Marwan Krunz, and Srinivasan Ramasubramanian. Node clustering in wireless sensor networks : recent developments and deployment challenges. *IEEE Network*, 20(3) :20–25, 2006.