Agrégation Distribuée de Données dans les Réseaux Dynamiques

Quentin Bramas, Toshimitsu Masuzawa, Sébastien Tixeuil

Agrégation Distribuée de Données dans les Réseaux Dynamiques †

Quentin Bramas[1,2], Toshimitsu Masuzawa[3] et Sébastien Tixeuil[1,2]

[1] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, F-75005, Paris, France*
[2] *CNRS, UMR 7606, LIP6, F-75005, Paris, France* [3] *Osaka University, Japan*

L'agrégation de données dans un réseau est un problème central pour de nombreuses applications. Il consiste a récupérer les données de chaque nœud du réseau, en supposant que deux données peuvent être agrégées en une seule, et que chaque nœud ne transmet une donnée qu'une seule fois. Des solutions distribuées performantes (du point de vue du délai) existent dans le cas d'un réseau statique, même en présence de collisions (comme dans les réseaux de capteurs sans fil). Cependant, dans le cas des réseaux dynamiques, le problème est NP-difficile, même avec un algorithme centralisé qui connait l'évolution future du réseau. Dans cet article, nous étudions le problème de l'agrégation de données dans des réseaux dynamiques par un algorithme distribué, dans le cas où il n'y a pas de collisions. Après avoir défini formellement le problème, nous prouvons qu'il n'est pas solvable sans connaissance supplémentaire, face à un adversaire sans mémoire, et ce même avec un algorithme probabiliste. Ensuite, nous étudions le problème face à un adversaire probabiliste choisissant les interactions dans le réseau de manière aléatoire. Dans ce cas nous donnons des algorithmes optimaux utilisant (i) une connaissance partielle du future ou (ii) aucune connaissance.

Mots-clefs : Agrégation de données, Réseaux Dynamiques, Algorithme Distribué

1 Introduction et Modèle

On considère un réseau dont chaque nœud est autonome et possède une donnée initiale. Chaque nœud a un identifiant et il existe un nœud particulier appelé le *puits*. Le but est de transmettre la donnée de chaque nœud au puits le plus rapidement possible. Cependant on suppose que chaque nœud ne peut transmettre une donnée détenue qu'une seule fois (par impossibilité de la dupliquer ou pour réduire la consommation d'énergie). Il est cependant possible à un nœud d'agréger une donnée reçue avec sa propre donnée (à l'aide d'une fonction d'agrégation, *e.g.* max, min, etc.). Le problème de l'*Agrégation de Données en Temps Minimum* consiste à agréger toutes les données du réseaux jusqu'au nœud puits en un minimum de temps, en autorisant une unique transmission par nœud.

Ce problème est central pour de nombreuses applications, *e.g.* des capteurs déployés dans une forêt qui doivent envoyer leur température à une station de base, des capteurs placés sur le corps humain qui doivent envoyer leur données vitales au téléphone portable du porteur, etc. Dans ces applications, le nœud puits joue un rôle de passerelle soit avec un autre réseau, soit avec un humain, qui seront capables d'exploiter les résultats.

Dans ce papier nous étudions le problème de l'agrégation de donnée distribuée dans le cas ou le réseau est dynamique, *i.e.*, lorsque l'ensemble des liens permettant la transmission de données entre les nœuds varie dans le temps. Notre modèle s'appuie sur le modèle des graphes évolutifs pour représenter l'évolution du réseau. Plus précisément, l'évolution de la topologie d'un réseau peut se modéliser par une séquence de graphes statiques, où chaque graphe représente l'état du réseau à un instant donné. Dans notre modèle, où les objets interagissent deux à deux, un adversaire, en plus de choisir la séquence de graphes statiques, choisit pour chaque graphe statique, l'ordre dans lequel ces interactions se produisent. C'est pourquoi nous supposons directement que chaque graphe de la séquence ne contient qu'un seul lien, c'est-à-dire que le

† Ce travail a été effectué dans le cadre du Labex SMART soutenu par les financements de l'état Français gérés par l'ANR sous le programme Investissements d'Avenir sous la référence ANR-11-IDEX-0004-02. Aussi, il est soutenu en partie par le LINCS.

réseau dynamique est modélisé par une séquence d'interactions (impliquant deux nœuds) représentant les interactions entre les nœuds au cours du temps. ‡

Travaux connexes. On peut classer les résultats existants sur l'agrégation de données en deux parties, selon qu'ils considèrent ou non que les collisions lors des transmissions sont prises en charge par la couche MAC ou pas. Dans le cas où les collisions ne sont pas gérées par la couche MAC, le problème est NP-complet [BT16], même dans des réseaux de capteurs statiques (resp. dynamiques) de degré maximum 3 (resp. 2). Dans le cas des réseaux de capteurs statiques, de nombreux travaux ont abouti à des algorithmes d'approximation distribués performants. Cependant, le seul algorithme d'approximation existant pour les réseaux dynamiques est centralisé et suppose la connaissance complète de l'évolution future du réseau.

Dans le cas où les collisions sont gérées par la couche MAC, on retiendra particulièrement les travaux de Cornejo et al. [CGN12] ou un ensemble de nœuds doit agréger des jetons. Aucun nœud ne joue le rôle de puits et le but est de minimiser le nombre de nœuds possédant au moins un jeton à la fin du temps imparti. Une solution pour laquelle un nœud particulier collecterait *tous* les jetons permettrait de résoudre le problème que nous considérons ici.

Modèle. Un réseau dynamique est modélisé par un couple (V,I) où V est l'ensemble des nœuds et $I = (I_t)_{t\in\mathbb{N}}$ est la séquence des interactions impliquant une paire de nœuds. Dans la suite, on notera n le nombre de nœuds et $s \in V$ le nœud puits.

On suppose que les nœuds ont un identifiant unique, une mémoire et une capacité de calcul illimitée. Cependant, on pourra considérer des nœuds sans mémoire persistante. Initialement, chaque nœud de V possède une donnée. Lors d'une interaction $I_t = \{u,v\}$, si à la fois u et v possèdent une donnée, l'un des nœuds peut transmettre sa donnée à l'autre, qui l'agrège avec sa propre donnée. Après avoir transmis sa donnée, un nœud ne peut plus ni recevoir de donnée, ni transmettre à nouveau.

Un algorithme distribué d'agrégation de données (DODA) est un algorithme qui, étant donnée une interaction $I_t = \{u,v\}$ et sa date d'occurrence t, retourne u, v, ou $\perp$. Si les deux nœuds u et v possèdent une donnée avant l'interaction, et si l'algorithme retourne un nœud (u ou v), alors ce dernier agrège la donnée de l'autre nœud. Si l'algorithme retourne $\perp$, aucune donnée n'est transmise. Lors de son exécution, un DODA peut utiliser sa mémoire interne, la modifier, et utiliser des informations concernant les nœuds. Par défaut, un nœud u a pour information son identifiant $u.ID$ et s'il est ou non le puits $u.isSink$. Certains algorithmes nécessitent d'autres informations (comme la séquence des interactions futures qui impliquent le nœud $u.future$, ou le moment de la prochaine interaction entre u et le puits $u.meetTime$). L'ensemble des DODA utilisant les informations supplémentaires $\mathrm{i}_1, \mathrm{i}_2, \ldots$ est noté $\mathcal{D}_{\mathsf{ODA}}(\mathrm{i}_1, \mathrm{i}_2, \ldots)$. Le sous ensemble de $\mathcal{D}_{\mathsf{ODA}}(\mathrm{i}_1, \mathrm{i}_2, \ldots)$ des algorithmes sans mémoire (*i.e.*, n'utilisant pas de mémoire persistante) est noté $\mathcal{D}^{\emptyset}_{\mathsf{ODA}}(\mathrm{i}_1, \mathrm{i}_2, \ldots)$

On suppose qu'un adversaire construit la séquence des interactions. On distingue plusieurs types d'adversaire : les adversaires sans mémoire (qui construisent la séquence avant le début de l'exécution), et les adversaires aléatoires (qui choisissent les interactions aléatoirement).

Par manque de place, dans ce papier nous considérons uniquement des séquences d'interactions $I = (I_t)_{t\in\mathbb{N}}$ telles que l'agrégation de données est toujours possible sur tout suffixe $(I_t)_{t>k}$ (au moins par un algorithme centralisé qui connait le futur). De telles séquences sont dites $\mathcal{D}_{\mathsf{ODA}}$-récurrentes. Dans ce contexte, on s'intéresse aux conditions d'existence d'un DODA qui termine pour toute séquence $\mathcal{D}_{\mathsf{ODA}}$-récurrente. Certaines preuves sont disponibles dans le rapport technique associé [BMT16].

2 Face à un Adversaire Sans Mémoire

Un adversaire sans mémoire connait le code de l'algorithme, mais il doit construire la séquence d'interactions sans connaitre les choix (possiblement probabilistes) effectués par l'algorithme. Cela revient à générer, avant même le début de l'exécution, l'intégralité de la séquence d'interactions. Dans le cas d'un algorithme déterministe, cet adversaire est équivalent à un adversaire pouvant s'adapter aux choix faits par l'algorithme (ces choix peuvent être prédits avant le début de l'exécution). Face à un algorithme probabiliste, la séquence de nombres aléatoires n'étant pas connue de l'adversaire, les choix de l'algorithme deviennent

‡. Cette modélisation des interactions se rapproche de celle utilisée pour les protocoles de populations, mais notre modèle de nœud et d'adversaire est très différent de ceux utilisés dans le contexte de protocoles de populations.

imprévisibles. Il est facile de trouver un adversaire empêchant tout algorithme déterministe d'agréger les données du réseau en temps fini. Notre premier théorème généralise ce résultat à tout algorithme probabiliste sans mémoire, laissant ouvert le cas des algorithmes probabilistes avec mémoire.

Théorème 2.1 *Pour tout algorithme probabiliste $A \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$, il existe un adversaire sans mémoire générant une séquence d'interactions $\mathcal{D}_{\mathsf{ODA}}$-récurrente I telle que l'exécution de A sur I ne termine pas, avec forte probabilité (avec un probabilité supérieure à $1-o(1/\log(n))$).*

Preuve: Soit $V = \{s, u_0, \ldots, u_{n-2}\}$. Dans la suite, les indices des nœuds sont modulo $n-1$, *i.e.*, pour tout $i \in \mathbb{N}$ $u_i = u_j$, avec $i \equiv j\,[n-1]$. Soit I^{∞} défini pour tout $i \in \mathbb{N}$ par $I^{\infty}_i = \{u_i, s\}$. Soit I^l le préfixe de longueur $l \in \mathbb{N}$ de la séquence I^{∞}. Pour tout $l \in \mathbb{N}$, l'adversaire peut calculer la probabilité P_l qu'aucun nœud ne transmette sa donnée pendant l'exécution de A sur I^l. La suite $(P_l)_{l>0}$ est décroissante et minorée, donc converge vers une limite $\mathcal{P} \geq 0$. Pour un l donné, si $P_l \geq 1/n$, alors on peut montrer qu'au moins deux nœuds ont une probabilité de ne pas transmettre supérieure à $n^{-\frac{1}{n-2}} = 1 - O(\frac{1}{n})$ (pendant l'exécution de A sur I^l). Donc si $\mathcal{P} \geq 1/n$, alors il existe un nœud qui ne transmet jamais, et l'exécution de A sur I^{∞} ne termine pas, avec forte probabilité.

Sinon, soit $l_0 \in \mathbb{N}$ le plus petit indice tel que $P_{l_0} < 1/n$. Avec forte probabilité, au moins un nœud transmet pendant l'exécution de A sur I^{l_0}. De plus, $P_{l_0-1} \geq 1/n$, donc deux nœuds u_d et $u_{d'}$ n'ont pas transmis pendant l'exécution de A sur I^{l_0-1}, avec une probabilité supérieure à $n^{-\frac{1}{n-2}}$ (si $l_0 = 0$, on peut choisir $\{u_d, u_{d'}\} = \{u_1, u_2\}$). On a $u_d \neq u_{l_0}$ ou bien $u_{d'} \neq u_{l_0}$. Sans perte de généralité, on suppose que $u_d \neq u_{l_0}$. Ainsi la probabilité que u_d transmette pendant l'exécution de A sur I^{l_0-1} est la même que sur I^{l_0}.

Soit I' la séquence finie d'interactions $(\{u_d, u_{d+1}\}, \{u_{d+1}, u_{d+2}\}, \ldots, \{u_{d-2}, u_{d-1}\}, \{u_{d-1}, s\})$. Cette séquence est construite de sorte que u_d doit envoyer sa donnée jusqu'à s en utilisant un chemin qui contient tous les autres nœuds. On construit alors la séquence d'interactions I comme étant la séquence I^{l_0} suivie de I' répétée une infinité de fois. Ainsi, après l'exécution de A sur I^{l_0}, avec forte probabilité, u_d contient toujours sa donnée et au moins un autre nœud a déjà transmit ça donnée. Donc, lors de l'exécution de A sur I', la donnée de u_d ne peut plus atteindre s (le chemin que doit emprunter la donnée de u_d contient tous les autres nœuds, dont au moins un qui a déjà transmis et ne peut plus transmettre). L'exécution de A sur I ne termine pas. Par ailleurs, comme la séquence I' est suffisante pour agréger toutes les données de V, I est $\mathcal{D}_{\mathsf{ODA}}$-récurrente. □

3 Face à un Adversaire Probabiliste

On considère maintenant un adversaire qui choisit les interactions aléatoirement, uniformément parmi toutes les interactions possibles. Chaque interaction $I = \{u, v\}$, $u, v \in V$, a une probabilité $\frac{2}{n(n-1)}$ d'être choisie par l'adversaire. Comme l'apparition d'une interaction ne dépend pas des interactions précédentes, les deux algorithmes que nous proposons dans cette section fonctionnent sans mémoire, *i.e.*, la sortie de nos algorithmes dépend uniquement de l'interaction et des informations concernant les nœuds impliqués. De plus, on suppose que les nœuds proposés à l'algorithme sont ordonnés selon leur identifiant, et qu'ils contiennent tous les deux une donnée (sinon l'algorithme retourne $\perp$).

— *Gathering* ($\mathcal{GA} \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$) : Un nœud transmet sa donnée lorsqu'il est connecté à s ou à tout autre nœud possédant une donnée. $\mathcal{GA}$ associe à (u_1, u_2, t) le nœud u_i si $u_i.isSink$, sinon il associe u_1.

— *Waiting Greedy* de paramètre $\tau \in \mathbb{N}$ ($\mathcal{WG}_{\tau} \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}(meetTime)$) : le nœud avec le plus grand $meetTime$ transmet si son $meetTime$ est plus grand que τ :

$$\mathcal{WG}_{\tau} : (u_1, u_2, t) = \begin{cases} u_1 & \text{si } m_1 < m_2 \wedge \tau < m_2 \\ u_2 & \text{si } m_1 > m_2 \wedge \tau < m_1 \\ \perp & \text{sinon} \end{cases} \qquad \begin{array}{l} m_1 = u_1.meetTime(t) \\ m_2 = u_2.meetTime(t) \end{array}$$

Bornes Inférieures. Nous étudions la durée minimum d'agrégation des données qu'il est possible d'obtenir. Sans connaissance supplémentaire, un algorithme est limité par la durée nécessaire pour agréger la donnée du dernier nœud (autre que s). En effet, lorsqu'il ne reste qu'un nœud $u \neq s$ possédant une donnée, la probabilité que u puisse transmettre sa donnée est égale à la probabilité d'occurrence de l'interaction $\{u, s\}$, *i.e.*, $\frac{2}{n(n-1)}$. L'espérance d'un tel événement est $\frac{n(n-1)}{2}$, d'où le théorème suivant :

Théorème 3.1 *Tout algorithme $A \in \mathcal{D}_{\mathsf{ODA}}()$ termine en moyenne en $\Omega(n^2)$ interactions.*

Pour les algorithmes utilisant des informations supplémentaires, nous allons minorer le nombre d'interactions par le nombre d'interactions requis par un algorithme centralisé connaissant toute la séquence d'interactions. Pour cela on peut remarquer que l'agrégation de données correspond à un broadcast en prenant le temps qui s'écoule dans le sens inverse. On peut montrer qu'un broadcast s'effectue avec forte probabilité en $\Theta(n\log(n))$ interactions, d'où le théorème suivant.

Théorème 3.2 *Tout algorithme optimal $A \in \mathcal{D}_{\mathsf{ODA}}(\texttt{séquence d'interactions})$ termine en $\Theta(n\log(n))$ interactions avec forte probabilité.*

Performance des Algorithmes. Pour l'algorithme $\mathcal{GA}$, on définit la v.a. X_i du nombre d'interactions entre la $(i-1)$-ème et la i-ème transmission. X_i suit une loi géométrique de paramètre $\frac{(n-i+1)(n-i)}{n(n-1)}$. La durée moyenne d'exécution de $\mathcal{GA}$ est donc la somme $\sum_1^{n-1} E(X_i) = n(n-1)\sum_{i=1}^{n-1} \frac{1}{i(i+1)} \in O(n^2)$. $\mathcal{GA}$ est donc optimal dans $\mathcal{D}_{\mathsf{ODA}}()$.

Pour les performances de $\mathcal{WG}_\tau$ on pose f une fonction telle que $f(n) = \omega(1)$ et $f(n) = o(n)$.

Théorème 3.3 *L'algorithme $\mathcal{WG}_\tau$ avec $\tau = nf(n) + n^2\log(n)/f(n)$ termine en τ interactions, avec forte probabilité.*

Preuve: Soit L l'ensemble des nœuds qui interagissent avec s entre les instants $\tau' = n^2\log(n)/f(n)$ et τ, et soit L^c son complémentaire (on suppose $s \in L$). Lors d'une interaction avant l'instant τ', *(i)* si les deux nœuds sont dans L, rien ne se produit (car leur *meetTime* est inférieur à τ), *(ii)* si les deux nœuds sont dans L^c, l'un des nœuds transmet à l'autre, *(iii)* sinon le nœud dans L^c transmet sa donnée au nœud dans L. Une manière possible d'agréger toutes les données est que chaque nœud dans L^c interagisse au moins une fois avec un nœud dans L avant l'instant τ'. Ensuite les nœuds de L transmettent directement au puits entre τ' et τ. On peut montrer qu'avec forte probabilité, $nf(n)$ interactions sont suffisantes pour que $f(n)$ nœuds interagissent avec le puits, donc L est de cardinal $f(n)$ avec forte probabilité. De plus, lorsque L contient au moins $f(n)$ nœuds, alors il faut au plus $n^2\log(n)/f(n)$ interactions pour que chaque nœud de L^c interagisse au moins une fois avec un nœud de L. Donc en choisissant $\tau = nf(n) + n^2\log(n)/f(n)$, l'algorithme termine avant l'instant τ avec forte probabilité. □

Le paramètre $\tau = 2n^{3/2}\sqrt{\log(n)}$, obtenu en prenant la fonction $f : n \mapsto \sqrt{n\log(n)}$ qui minimise la valeur de $nf(n) + n^2\log(n)/f(n)$, semble donc être optimal pour l'algorithme $\mathcal{WG}_\tau$. On peut montrer que c'est le cas, et même que l'algorithme correspondant $\mathcal{WG}_\tau$, avec $\tau = 2n^{3/2}\sqrt{\log(n)}$, est asymptotiquement optimal parmi tout les algorithmes dans $\mathcal{D}_{\mathsf{ODA}}(meetTime)$ (même ceux avec mémoire).

4 Conclusion

Nous présentons un nouveau modèle pour l'étude du problème de l'agrégation de données distribué dans les réseaux dynamiques. Après avoir présenté le résultat d'impossibilité dans le cas général, face à un adversaire sans mémoire, nous donnons deux algorithmes optimaux face à un adversaire aléatoire, le premier fonctionnant sans aucune connaissance supplémentaire, le deuxième utilisant l'information *meetSink*.

Références

[BMT16] Quentin Bramas, Toshimitsu Masuzawa, and Sébastien Tixeuil. Distributed Online Data Aggregation in Dynamic Graphs. Research report, Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris. ; Osaka University, Japan, January 2016.

[BT16] Quentin Bramas and Sébastien Tixeuil. The complexity of data aggregation in static and dynamic wireless sensor networks. *Information and Computation*, pages –, 2016.

[CGN12] Alejandro Cornejo, Seth Gilbert, and Calvin Newport. Aggregation in dynamic networks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 195–204. ACM, 2012.