Diffusion fiable dans les réseaux dynamiques

Guillaume Béduneau, Bertrand Ducourthial

Diffusion fiable dans les réseaux dynamiques

Guillaume Béduneau et Bertrand Ducourthial

Sorbonne universités,
Université de technologie de Compiègne
CNRS, UMR 7253 Heudiasyc
CS 60319 - 60203 Compiègne cedex, France

Nous présentons un algorithme réparti capable de réaliser un échange total (*gossiping*) dans un réseau dynamique. Le but est de diffuser les messages de tous les mobiles malgré leur mémoire limitée et les déconnexions du réseau.

Mots-clefs : Réseaux dynamiques, réseaux de robots, réseaux véhiculaires, diffusion fiable, *gossiping*

1 Introduction

De nombreux travaux s'intéressent aux flottes de *mobiles* (véhicules, robots, drones...). En effet, la démocratisation des équipements permet d'envisager de nouvelles applications, dans lesquelles les mobiles coopèrent pour réaliser une tâche commune. Dans ces flottes, il n'est pas toujours possible ni efficace d'envisager des communications centralisées, via un point d'accès partagé par tous les mobiles. Par exemple, le WiFi véhiculaire IEEE 802.11p permet des communications véhicule-véhicule pour réduire la latence. De même, le futur standard de réseau mobile 5G permet des communications directes de mobile à mobile sans passer par la station de base. Les applications coopératives reposent alors sur des réseaux de véhicules, de drones ou de robots, qui se distinguent des autres par leur dynamique, qu'elle soit subie (véhicules) ou contrôlable par l'utilisateur (robots, drones).

Les mobiles sont généralement équipés de capteurs et de calculateurs, qui produisent régulièrement de nombreuses données. Or, ces données sont utiles aux autres participants pour élaborer une information commune ou réaliser une tâche coopérative. Dans ce contexte, nous nous intéressons à un algorithme réparti permettant de réaliser un échange total (*gossiping* [Rum94]) dans le réseau dynamique : chaque nœud mobile émet périodiquement un message, qui doit être reçu par tous les autres mobiles malgré les pertes et déconnexions.

Si cet échange total est simple à mettre en oeuvre dans un réseau fixe, ce n'est pas le cas dans un réseau dynamique où la topologie est mouvante voire temporairement déconnectée et les pertes de messages nombreuses. Pour pallier ces difficultés, les messages émis doivent être stockés par les mobiles tant qu'ils n'ont pas la certitude que chaque participant les aient reçus. La gestion de ces *caches* devient alors critique.

Parmi les solutions proposées dans la littérature, on trouve des techniques reposant sur des structures topologiques [Seg83, Lim01]. Cependant, elles ne peuvent supporter une forte dynamique. La plupart des travaux s'intéressent à la minimisation du nombre de messages, au délai de diffusion ou aux conditions permettant de supporter des pannes [Bon18]. Mais peu d'entre eux s'intéressent à la taille des caches. Nous proposons un algorithme réparti réalisant une diffusion fiable dans les réseaux dynamiques et nous étudions différentes stratégies pour réduire à la fois le délai de diffusion et la taille des caches.

2 Algorithme

Diffusion. Dans un réseau fiable de topologie inconnue mais fixe, la diffusion est un problème classique [Seg83]. Chaque nœud du réseau retransmet une seule fois tout nouveau message reçu. Mais dans un réseau sans fil, cette stratégie peut amener à des collisions et donc à une absence de réception par certains noeuds. En outre, si la topologie est dynamique, il est possible qu'un nœud n'ayant pas encore reçu le message se

déplace dans une zone du réseau où tous ses nouveaux voisins l'ont déjà retransmis ; il ne le recevra donc jamais. Il est donc nécessaire d'adjoindre un mécanisme de retransmission.

Demande de retransmission. Dans le cadre d'une diffusion, les acquittements ne sont guère utilisables et la détection des non réceptions revient aux destinataires, qui comparent pour cela des numéros de séquence. En cas de déséquencement (et donc de perte), un nœud pourrait avertir l'émetteur du dernier message dans son voisinage. Mais rien ne dit que celui-ci est encore à portée radio ni qu'il a lui-même reçu le message manquant. Il est donc préférable d'adresser la requête à tous les voisins. Cependant, dans un réseau dynamique, cette méthode risque de générer de trop nombreuses requêtes et donc de surcharger le réseau. Or, un nœud détecte une perte lorsqu'il reçoit un message, qu'il devra certainement rediffuser (s'il est nouveau pour lui). Il est donc plus efficace d'adjoindre la demande de retransmission au message rediffusé (*piggybacking*) lorsqu'il y a rediffusion [†].

Définissons maintenant le contenu de la demande de retransmission. Puisqu'elle est adressée à son voisinage, si le nœud indique uniquement le premier message manqué, plusieurs voisins pourraient le lui envoyer, ce qui limiterait l'efficacité des retransmissions tout en surchargeant le réseau. En fait, à ne redemander qu'un seul message manqué par nouveau message rediffusé, il se pourrait que l'algorithme ne converge pas selon l'état du réseau. À l'inverse, si le demandeur mentionne tous les messages manqués, alors la taille des messages devient non bornée. Pour conserver une taille de message constante, plutôt que d'indiquer les messages à retransmettre, il est préférable d'indiquer les messages qu'il est inutile de retransmettre. En effet, cela peut se faire avec le plus grand numéro de séquence des messages consécutifs reçus de chaque émetteur [‡]. La taille de message résultante est en $O(n)$ où n est le nombre de nœuds du réseau. Si n n'est pas trop grand, cette taille de message est acceptable. Pour n grand, il est préférable de n'envoyer que certains numéros de séquence, choisis aléatoirement ou à tour de rôle.

Retransmission des messages perdus. Afin de répondre aux demandes de retransmission, chaque nœud garde en cache les messages qu'il a reçus. Pour éviter que les caches ne croissent indéfiniement, les nœuds les éliminent une fois leur durée de vie écoulée ou bien lorsqu'ils ont été reçus par l'ensemble des nœuds (il n'est pas possible de prévoir qui pourra rencontrer le nœud ayant manqué des messages).

Pour assurer une diffusion fiable en mémoire limitée, il est donc primordial de déterminer quand un message a été reçu par tous les participants. L'information de bonne réception doit donc se propager dans le réseau. Or, pour l'instant chaque nœud ne diffuse que son propre état à ses voisins. Pour qu'il y ait propagation de cet état à plus d'un saut, il est nécessaire qu'un nœud propage l'état de ses voisins. Comme précédemment, on utilise le *piggybacking* pour ne pas multiplier les messages. Pour conserver un champ de contrôle de taille raisonnable (l'état des nœuds atteint une taille en $O(n^2)$), là-encore une partie seulement de l'information est jointe au message. Finalement, pour éviter d'attendre la réception de messages par des nœuds qui ont disparu du réseau, un délai de garde est utilisé [§].

Reste à voir quel message un nœud retransmettra à son voisin s'il constate qu'il en a manqué en comparant son cache avec l'état reçu. Plusieurs stratégies ont été envisagées. Avec la stratégie *old*, le plus ancien message non reçu est retransmis. L'état du voisin progresse régulièrement au risque de voir de nombreux doublons dans les retransmissions car les voisins enverront le même message manquant. La stratégie *rand* choisit au hasard un message à renvoyer, dans le but d'éviter les doublons dans les retransmissions [Bel11]. Cependant, de part l'information limitée dont on dispose sur le voisin (numéro du premier message manqué), il se pourrait qu'un message déjà reçu soit renvoyé. En outre, l'état renvoyé par le voisin pourrait être identique. La stratégie *mix* combine les deux premières dans le but d'atténuer leurs inconvénients respectifs. La stratégie *exp* consiste à effectuer un tirage aléatoire privilégiant les plus anciens messages. L'idée est de faire rapidement évoluer l'état du voisin tout en limitant les doublons.

Intuition de preuve. Tout algorithme réparti est susceptible d'échouer si la dynamique est trop forte. Pour s'en convaincre, il suffit de considérer un système dans lequel la durée de vie des liens de communication

†. En l'absence de nouveau message, un *timer* permet de vérifier auprès des voisins qu'aucun message n'a été manqué.

‡. Par exemple, si les messages 1 à 10 puis 12 à 14 ont été reçus du nœud 1 d'une part et les messages 1 à 5 puis 7 à 8 du nœud 2 d'autre part, l'état du récepteur est donné par $[10, 5]$.

§. Un nœud est considéré définitivement perdu s'il n'a pas donné signe de vie avant l'expiration du délai de garde.

est inférieure au délai nécessaire à l'acheminement d'un fragment de message. Nous posons donc des conditions sur la dynamique [Duc16].

On appelle 1-graphe le graphe modélisant le réseau de sorte que chaque arête permette l'émission d'un message. Comme l'on suppose des diffusions périodiques par les nœuds, la durée de vie du lien de communication est supposée supérieure ou égale au délai inter-messages d. On montre ainsi que si le 1-graphe modélisant la topologie reste connexe, alors la diffusion est accomplie en temps fini. Le fait que le 1-graphe reste connexe n'implique pas que la topologie soit fixe ; le délai de diffusion est donc estimé expérimentalement.

En cas de déconnexion temporaire d'un nœud, et uniquement si la durée de déconnexion est inférieure au délai de garde, on montre que le nœud obtient les messages manquant en temps fini après sa reconnexion.

La taille du cache est supérieure ou égale à tn/d, où n est le nombre de nœuds et t la durée moyenne de stockage en cache. Cette durée dépend du délai de diffusion des messages et de l'état des sites ; elle est donc elle aussi estimée expérimentalement.

3 Éxpérimentations et mesure de performances

Programmation. Afin de mener des expérimentations, l'algorithme décrit précédemment a été programmé en Python dans l'intergiciel Airplug dédié à l'étude des réseaux dynamiques ; des expérimentations ont ensuite été menées par émulation de réseau [Bui10].

Comme n est souvent borné dans une flotte de robots, nous avons considéré qu'il n'était pas trop grand et nous envoyons donc l'état complet du nœud avec le message à diffuser. Par contre, l'état d'un seul autre noeud (choisi au hasard) est retransmis dans les messages (toujours en *piggybacking*), afin de le propager à plusieurs sauts. Ces compromis impactent la taille des messages mais aussi la propagation des états des nœuds et donc le temps de stockage des messages en cache (t). Par ailleurs, la durée de vie des messages est choisie supérieure au délai de garde des nœuds.

Une fois l'algorithme programmé, une preuve de concept sur robots terrestres a été réalisée. L'application consistait à diffuser des ordres de pilotage. Alors que des pertes de messages ont été constatées en l'absence de la diffusion fiable, tous les messages étaient reçus lorsqu'elle était employée.

Expérimentations. Pour évaluer les performances de notre algorithme, nous avons utilisé quatres métriques :

- **r**essources nécessaires :
  - **rr** (réseau) : nombre total de messages émis au cours d'une exécution ;
  - **rm** (mémoire) : nombre maximal de messages en cache au cours d'une exécution.
- Intérêt **a**pplicatif :
  - **ad** (délai) : durée moyenne de transmission ;
  - **ae** (efficacité) : nombre de messages reçus au niveau applicatif au cours de l'exécution.

Les scénarios utilisés consistaient en des convois de robots plus ou moins rapprochés, impactant donc le degré des nœuds et le diamètre du réseau (la portée de communication étant fixe). L'influence du taux de pertes et du délai d'attente avant retransmission d'un message ont été étudiés en conjonction avec la stratégie de retransmission.

Résultats. Par manque de place, nous ne présentons qu'une partie des résultats, dans le cas du scénario défavorable d'une topologie en chaîne. La figure 1 permet de comparer — avec les métriques ae et rm — les stratégies de sélection des paquets à retransmettre pour différents taux de perte dans le réseau.

On peut remarquer sur la figure de gauche que la stratégie *old* (qui privilégie le plus ancien message non reçu) est la moins efficace car elle ne permet pas de diffuser autant de messages que les autres. Pourtant, son inconvénient majeur (doublons dans les retransmissions) devrait être atténué dans ce scénario où le degré est faible. Par contre, les stratégies aléatoires ou partiellement aléatoires s'avèrent meilleures en terme d'efficacité de l'algorithme, y compris lorsque le degré est faible.

La figure de droite donne la taille du cache pour les différentes stratégies. Les bons résultats de la stratégie *old* sont trompeurs car cette stratégie parvient à diffuser bien moins de messages que les autres, comme le montre le graphique de gauche. On constate que la stratégie *exp* est la plus efficace des trois stratégies aléatoires en terme de taille du cache. À noter que la décroissance des tailles du cache pour des très forts

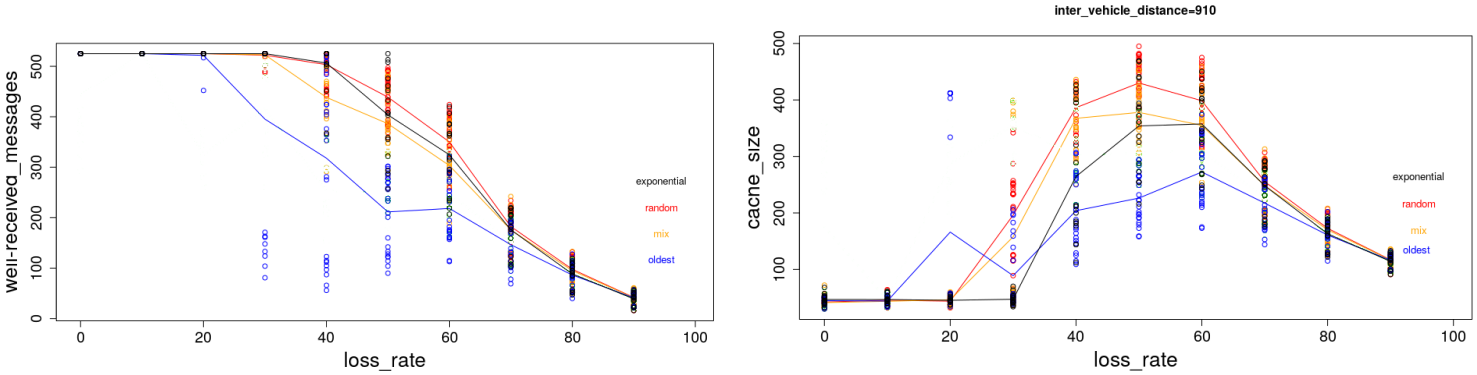

FIGURE 1: Métriques d'efficacité applicative **ae** (gauche) et de consommation de ressources **rm** (droite) en fonction du taux de perte

taux de perte s'explique par le fait que, dans ces conditions extrêmes, peu de messages sont transmis et donc peu sont stockés.

Pour conclure, un tirage aléatoire des messages à retransmettre s'avère préférable au choix du premier manquant. Mais un tirage uniforme est moins efficace qu'un tirage privilégiant les plus anciens messages.

4 Conclusions

Dans cet article, nous nous sommes intéressés à la diffusion fiable dans les réseaux dynamiques, tels que les véhicules, les robots, les drones... Il s'agit de permettre aux mobiles de transmettre efficacement leurs données à tous les autres (échange total) sans consommer trop de ressources réseaux et mémoire (cache).

Nous avons proposé un algorithme qui peut employer diverses stratégies pour le renvoi des paquets perdus. Cette solution fonctionnelle a été testée sur une flotte de huit robots. L'étude par émulation de réseau montre qu'une stratégie basée sur un tirage aléatoire qui privilégie les paquets anciens est préférable.

Les travaux futurs porteront sur l'amélioration de l'algorithme. Les résultats présentés sont liés à l'état du nœud transmis aux voisins ; des progrès peuvent être espérés si une information plus riche sans être trop grande était transmise. La recherche de métriques plus adaptées (métriques normalisées ou adimensionnelles) pourrait également faciliter les comparaisons.

Références

[Bel11] N. Belblidia, M. Dias De Amorim, L.H.M. Kosmalski, J. Leguay et V. Conan. PACS : Chopping and shuffling large contents for faster opportunistic dissemination. In IEEE WONS 2011 pages 9–16, Bardonecchia, Italy, January 2011.

[Bui10] A. Buisset, B. Ducourthial, F. El Ali et S. Khalfallah. Vehicular networks emulation. In IEEE ICCCN 2010, Zurich, Switzerland, August 2010.

[Bon18] S. Bonomi, G. Farina et S. Tixeuil. Reliable broadcast in dynamic networks with locally bounded byzantine failures, preprint, 2018.

[Rum94] J. de Rumeur. *Communication dans les réseaux de processeurs*. Masson, Paris, 1994.

[Duc16] B. Ducourthial et A.M. Wade. Dynamic p-graphs for capturing the dynamics of distributed systems. *Ad Hoc Networks*, 50 :13 – 22, 2016.

[Lim01] H. Lim et C. Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3) :353 – 363, 2001.

[Seg83] A. Segall. Distributed network protocols. *Information Theory, IEEE Transactions on*, 29 :23 – 35, 02 1983.