Sur un filtrage de latences pour du routage par renforcement sur des réseaux sans-fil

Alexis Bitaillou, Benoît Parrein, Guillaume Andrieux, Killian Couty

Sur un filtrage de latences pour du routage par renforcement sur des réseaux sans-fil

Alexis Bitaillou[1] †, Benoît Parrein[1] , Guillaume Andrieux[2] et Killian Couty[1]

[1] *Université de Nantes / LS2N, Nantes, France*
[2] *Université de Nantes / IETR, Nantes, France*

Q-routing est un algorithme de routage inspiré de Q-learning, un algorithme d'apprentissage par renforcement. Il utilise la latence comme métrique de routage. Mais la latence peut être difficile à mesurer correctement, en particulier dans les environnements radio des réseaux sans-fil. Dans ce papier, nous proposons d'ajouter un filtrage sur la mesure de la latence, afin d'améliorer son estimation. Nous comparons notre modification à Q-routing original. Dans notre scénario avec mobilité, le nombre de changements de route est divisé en moyenne par deux. Les gains en termes de qualité de service sont surtout dépendant de la vitesse des nœuds. Ces améliorations sont obtenues sans pour autant sacrifier la simplicité de Q-routing original.

Mots-clefs : Routage, Q-routing, réseaux sans-fil, apprentissage par renforcement

1 Introduction

Q-routing [BL94] utilise la latence comme métrique de routage. Dans leur article, Boyan et Littman montrent que Q-routing offre de bonnes performances en comparaison d'un algorithme utilisant le nombre de sauts. Q-routing a été évalué sur un simulateur personnel, les hypothèses de fonctionnement ne sont pas connues malheureusement. Même si Q-routing a été implémenté sur différents simulateurs réseaux [HMS08, HCL18], le problème n'est jamais mentionné même s'il est de toute évidence présent.

Dans cet article, nous montrons que la mesure de latence peut être bruitée sur un simulateur. Cela peut mener Q-routing à prendre des décisions sur une mauvaise estimation de la latence. Nous proposons d'ajouter une méthode de filtrage afin d'améliorer l'estimation de la latence. Nous proposons d'ajouter aussi une méthode afin de diminuer l'influence de la valeur d'initialisation de la Q-valeur. Nous évaluons la pertinence de nos modifications par rapport à la version originale de Q-routing sur un scénario.

Cet article est organisé comme suit. Dans la Section 2, nous verrons quelques précédentes contributions liées à Q-routing. La Section 3 détaillera nos modifications apportées à Q-routing. La Section 4 portera sur la description de l'expérimentation ainsi que les résultats obtenus. Enfin, une conclusion terminera cet article. L'implémentation de Q-routing et les scénarios sont disponibles sur notre dépôt Git.

2 Etat de l'art

2.1 Q-routing

Q-routing [BL94] est un algorithme de routage conçu par Boyan et Littman en 1994. Il s'inspire de Q-learning [WD92], un algorithme d'apprentissage par renforcement conçu deux ans plus tôt. Dans cet algorithme, chaque nœud x cherche la route vers d passant par y ayant la plus petite Q-valeur. La Q-valeur étant définie par la Q-fonction :

$$\Delta Q_x(d,y) = \eta(q+s+t-Q_x(d,y)) \tag{1}$$

†Issue du projet COWIN supporté par les RFI WISE et RFI Atlanstic 2020

où η est le taux d'apprentissage, q la durée passée dans la file d'attente, s la durée de transmission et t est la Q-valeur minimale des voisins y vers d. Seules les Q-valeurs des routes choisies sont actualisées. Dans leurs travaux, Boyan et Littman ont comparé Q-routing à l'algorithme de plus court chemin de Bellman-Ford. À faible charge réseau, Q-routing ne trouve pas le chemin le plus court et est donc plus lent. Cependant, Q-routing équilibre la charge entre les deux chemins disponibles et permet donc de limiter la congestion à forte charge. Néanmoins, ces résultats ont été obtenus sur leur simulateur ce qui rend leur simulation difficilement reproductible. Plusieurs adaptations de Q-routing ont été proposées dans la littérature.

AQ-routing Q-routing n'a pas été conçu pour les scénarios avec de la mobilité. Serhani *et al.* [SNJ20] proposent Adaptive Q-routing (AQ-routing), une extension de Q-routing adaptée à la mobilité. AQ-routing reprend certains concepts d'OLSR. AQ-routing utilise une Q-fonction complexe s'appuyant sur la stabilité plutôt que la latence. Dans leur article, ils comparent AQ-routing à OLSR (*Optimized Link State Routing*, standard et ETX) sur le simulateur *ns-3*. Sur un cas statique, AQ-routing offre un taux de paquets livrés similaire à OLSR-ETX mais il achemine les paquets plus lentement. En mobilité, il achemine le plus de paquets mais ce n'est qu'à partir de 4 m/s que le délai d'AQ-routing devient meilleur que celui d'OLSR-ETX. Pour obtenir ces performances, Serhani *et al.* ont augmenté la complexité de la Q-fonction utilisant la métrique ETX ainsi que d'autres coefficients.

Reinforcement Learning for LT Optimisation Q-routing n'est pas conçu spécifiquement pour les réseaux sans-fil de capteurs (*Wireless Network Sensor*, WSN). Bouzid *et al.* [BSRO20] proposent *Reinforcement Learning for LT Optimisation* (R2LTO), une version de Q-routing adaptée aux réseaux de capteurs. Leur changement porte essentiellement sur la récompense. Ils utilisent l'énergie pondérée par le nombre de sauts. Ils ont comparé leur algorithme à Q-routing et RLBR, un autre algorithme spécialisé, sur leur simulateur. D'après leur simulation, R2LTO améliore la durée de vie du réseau jusqu'à 25 %. Les autres métriques de qualité de service ne sont pas présentées. De plus, les simulations portent uniquement sur l'algorithme. Le surcoût lié à la gestion du routage ne sont donc pas évalué.

Q^2-routing La version originale de Q-routing considère uniquement la latence. Hendriks *et al.* [HCL18] proposent une meilleure prise en charge des contraintes de qualité de service avec Q^2-routing. La Q-fonction de Q^2-routing comprend 3 coefficients supplémentaires dépendant des exigences de l'utilisateur en termes de qualité de service (taux de paquets livrés, délai de livraison et gigue). D'après leurs simulations sur le simulateur *ns-3*, Q^2-routing surpasse AODV et Q-routing sur les 3 métriques de qualité de service.

Conclusion Ces 3 exemples de modifications portent essentiellement sur la définition de la Q-fonction ou de la récompense. AQ-routing et R2LTO vont plus loin en enlevant la latence. Ces solutions contournent simplement le problème de la qualité de la mesure de la latence. Nous pensons qu'il est possible d'améliorer Q-routing sans pour autant modifier la Q-fonction, grâce à une meilleure estimation de la latence.

3 Détails d'implémentation

Vue d'ensemble de l'implémentation Notre implémentation de Q-routing est minimale. Il n'y a pas de fonction auxiliaire comme une table de voisinage. Le nombre de sauts maximum est 16. Le délai d'expiration des routes de 60 s pour les cas statiques, 10 s pour les scénarios avec mobilité. Les mises à jour des routes s'effectuent toutes les 10 s pour les cas statiques, 5 s en cas de mobilité. L'implémentation est entièrement distribuée. Les nœuds n'ont accès qu'aux informations locales. Plus de détails sont fournis dans [BPA21]. Notre implémentation est disponible sur notre dépôt Git [‡].

Cycle de vie des routes Nous avons modifié le cycle de vie des routes. Les routes ont deux états supplémentaires : "vérifiée" et "en probation". Cette modification permet de réduire le nombre de changements lié à la valeur d'initialisation et la pénalité liée à l'expiration. Lorsque qu'une route est ajoutée, elle est considérée comme "en probation". Une route devient "vérifiée" si elle reçoit au moins 3 mises à jour. Si la route expire, elle perd son statut "vérifiée". Les routes vérifiées sont choisies en priorité. S'il n'y a pas de route vérifiée, une route est choisie aléatoirement parmi les routes "en probation". Sans cette modification,

‡. https ://gitlab.univ-nantes.fr/ls2n-rio/qrouting-qualnet

dès qu'une route est ajoutée, elle est forcément à cause de la valeur d'initialisation, indépendamment de sa qualité réelle.

Filtrage La méthode de filtrage est une simple moyenne glissante. La latence moyenne est obtenue avec au maximum les 7 dernières mesures de latence. Les Q-valeurs sont donc calculées avec la latence moyenne au lieu de la dernière mesure de latence. Lorsqu'une route expire, les mesures de latences sont supprimées.

4 Expérimentation et résultats

Dans cette section, nous décrivons les différents éléments liés aux simulations ainsi que les résultats obtenus. Nous n'allons utiliser qu'une topologie simple illustrée par la Figure 1. Dans ce scénario, le nœud 9 envoie des données au nœud 10. Le flux de données a un débit constant (CBR) de 1 Mb/s. Le nœud 9 se déplace d'Est en Ouest selon un mouvement rectiligne uniforme. La distance à parcourir est de 1350 m. Il commence son déplacement 35 s après le début de la simulation. La simulation se termine 2 minutes après l'arrivée du nœud 9. Le tableau 1 récapitule certains paramètres de simulation. Les simulations sont répétées avec 25 graines différentes pour chaque combinaison de paramètres.

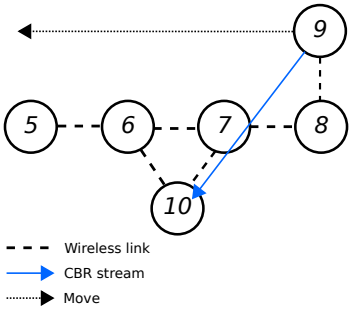


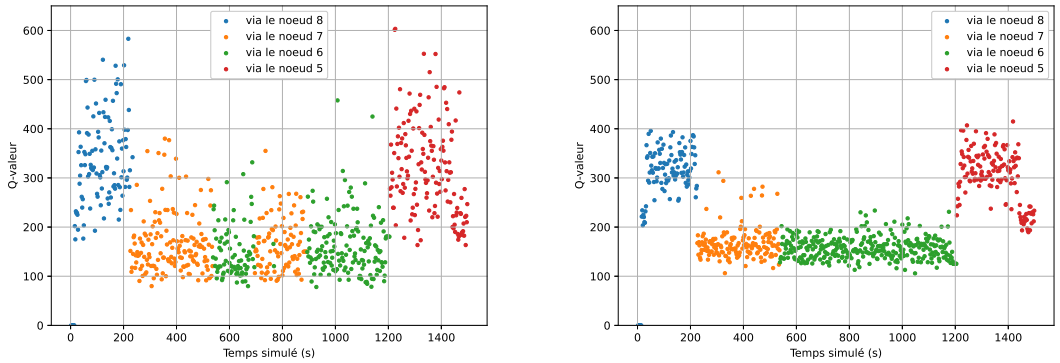
FIGURE 1 – Topologie de test.

	Paramètres	Valeur
Réseaux	Lien	IEEE 802.11a
	couche liaison	IEEE 802.11e
Nœud	File d'attente	1 file FIFO
Flux CBR	taille des messages	512 octets
Q-routing	Exploration	15 s
	η	0.9
	fenêtre de filtrage	7 échantillons

TABLE 1 – Paramètres de simulation

4.1 Résultats

Les Figures 2 montrent la pertinence du filtrage. Sans le filtrage, en raison du bruit de mesure et de l'équidistance entre les nœuds 6 et 7, Q-routing alterne entre les deux nœuds (entre 550 s et 900 s). Avec le filtrage, les Q-valeurs prennent des valeurs sur un intervalle plus petit. Il y a un moins de changement de route. En moyenne, le nombre de changements de route est divisé par deux (de 6 à 3 changements).



(a) Q-valeur en fonction du temps, sans filtrage.

(b) Q-valeur en fonction du temps, avec filtrage.

FIGURE 2 – Effet du filtrage sur les Q-valeurs.

Les résultats en termes de paquets livrés et de délai sont similaires pour Q-routing et sa version modifiée jusqu'à 4 m/s. Au-delà, Q-routing avec filtrage permet un léger gain en taux paquets livrés et en délai par rapport à l'original. Cette tendance est confirmée sur des topologies plus complexes que nous ne pouvons détailler ici pour des raisons de place.

4.2 Discussion

La fenêtre de filtrage peut être optimisée pour une topologie spécifique. Nous avons choisi une fenêtre de 7 échantillons car elle permet de meilleures performances en termes de qualité de service que les tailles de filtre inférieures. Cela représente aussi un compromis entre la qualité de filtrage et l'occupation en mémoire des échantillons. Néanmoins, le nombre de changements de route optimal est 3. Le nombre de changements de route moyen est parfois inférieur à 3 avec Q-routing avec filtrage. Il semble donc qu'un effet mémoire apparaît à cause de filtrage. Q-routing choisit plus longtemps les routes. L'amélioration du taux de paquets livrés et du délai en fonction de la vitesse semble confirmer cette hypothèse. Notre technique de filtrage est une simple moyenne glissante qui permet de ne pas augmenter significativement la complexité de Q-routing.

5 Conclusion

Dans cet article, nous avons proposé d'ajouter une méthode de filtrage sur la mesure de la latence de Q-routing. Nous l'avons évalué cet apport sur un scénario simple avec mobilité. Q-routing avec filtrage permet une diminution de moitié du nombre de changements de route sur notre scénario. De plus, il permet un léger gain en qualité de service en fonction la vitesse. Le nombre de changements de route montre qu'il est possible d'améliorer encore les performances en adaptant la taille du filtre en fonction du scénario.

Références

[BL94] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks : A reinforcement learning approach. In *Advances in neural information processing systems*, pages 671–678, 1994.

[BPA21] Alexis Bitaillou, Benoît Parrein, and Guillaume Andrieux. New Results on Q-Routing Protocol for Wireless Networks. In Luca Foschini and Mohamed El Kamili, editors, *Ad Hoc Networks*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 29–43, Cham, 2021. Springer International Publishing.

[BSRO20] S. E. Bouzid, Y. Serrestou, K. Raoof, and M. N. Omri. Efficient Routing Protocol for Wireless Sensor Network based on Reinforcement Learning. In *2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pages 1–5, September 2020. ISSN : 2687-878X.

[HCL18] Thomas Hendriks, Miguel Camelo, and Steven Latré. Q2-Routing : A Qos-aware Q-Routing algorithm for Wireless Ad Hoc Networks. In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 108–115, October 2018. ISSN : 2160-4886.

[HMS08] S. Hoceini, A. Mellouk, and B. Smail. Average-Bandwidth Delay Q-Routing Adaptive Algorithm. In *2008 IEEE International Conference on Communications*, pages 1840–1844, May 2008.

[SNJ20] Abdellatif Serhani, Najib Naja, and Abdellah Jamali. AQ-Routing : mobility-, stability-aware adaptive routing protocol for data routing in MANET–IoT systems. *Cluster Computing*, 23(1) :13–27, March 2020.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3) :279–292, May 1992.