Désynchronisation bio-inspirée dans les réseaux véhiculaires : quand les véhicules imitent les grenouilles

Alexandre Mouradian et Véronique Vèque

Laboratoire des Signaux et Systèmes (L2S, UMR8506), Université Paris Sud-CNRS-CentraleSupélec, Université Paris-Saclay F-91192 Gif-sur-Yvette
{*alexandre.mouradian, veronique.veque*}*@u-psud.fr*

A priori, on se dit qu'il n'est de chose plus éloignée d'un véhicule qu'une grenouille. Nous allons pourtant montrer dans cet article, que ces joyeuses bêtes ne sont pas seulement de passives victimes de nos routes, souvent réduites à deux dimensions entre gomme et asphalte. Ces créatures peuvent également nous servir de source d'inspiration pour développer des algorithmes permettant, sinon d'arrêter ce génocide batracien, d'au moins limiter le nombre de victimes, bien humaines elles, sur les routes. En effet, la sécurité routière est la principale motivation avancée pour le développement des réseaux véhiculaires. Cependant, de nombreux problèmes restent à résoudre pour que les applications de sécurité routière soient vraiment efficaces. Ces applications sont basées sur des échanges de messages périodiques appelés *beacons*. Ces *beacons* permettent aux véhicules d'avoir une vision élargie de leur environnement, et ainsi de pouvoir prévenir les situations dangereuses. Un des problèmes majeurs pour l'échange de *beacons* est que, dans les réseaux véhiculaires denses, les collisions de ces messages empêchent les véhicules d'avoir une vision à jour de leur environnement. Dans cet article, nous évaluons des algorithmes inspirés des désynchronisations de croassement de grenouilles qui permettent de réduire les collisions de manière distribuée et donc permettent aux véhicules d'avoir une meilleure vision de leur environnement.

Mots-clefs : VANET, Beaconing, Algorithme Distribué, Synchronisation en Opposition de Phase

1 Introduction

Une des motivations principales pour le développement de réseaux véhiculaires et de manière plus générale, de systèmes de transports intelligents (ITS), est l'amélioration de la sécurité routière. Les applications de sécurité routière reposent sur la possibilité d'échanger périodiquement des informations entre les véhicules. Des messages appelés *beacons* ou CAM (Cooperative Awarness Messages) [HL10] contentant a minima sa position, sa vitesse, sa direction et son accélération sont donc émis périodiquement par chaque véhicule. La réception de ces messages permet à un véhicule d'avoir une vision de son environnement à portée de communication. Le problème principal de ce mécanisme est que les collisions successives de messages peuvent empêcher certains véhicules d'avoir une vision à jour de leur environnement et donc de prévoir efficacement les situations dangereuses. À l'heure actuelle, les standards ITS prévoient que les *beacons* soient périodiquement diffusés en utilisant une variante du standard IEEE 802.11 (l'amendement 802.11p) [HL10]. Des travaux antérieurs ont mis en évidence le fait que la diffusion avec IEEE 802.11 n'est pas fiable [SCB11], spécialement dans les réseaux denses à cause des collisions non résolues. Ce problème induit des délais inter-*beacon* inacceptables pour les applications de sécurité routière.

Dans cet article, nous proposons d'étudier l'application d'algorithmes de désynchronisation bio-inspirés à la diffusion de *beacons* dans les réseaux véhiculaires. En effet, une solution pour réduire les collisions tout en conservant l'architecture ITS (notamment 802.11p) est de décaler les instants d'émission au niveau applicatif en utilisant de tels algorithmes.

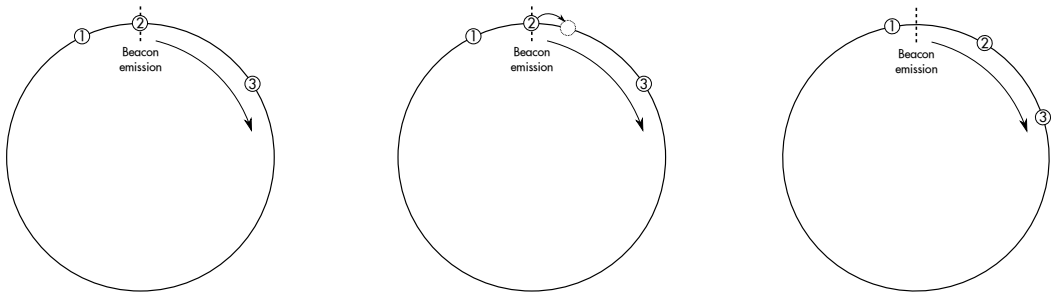
Les synchronisations spontanées (qui peuvent également être vus comme des consensus) peuvent être observées dans de nombreuses circonstances dans la nature [Str03]. On peut citer la synchronisation des

flashes des lucioles en Asie du Sud, le mouvement coopératif de bactéries ou de nuées d'oiseaux et les cellules cardiaques qui battent à l'unisson comme autant d'exemple d'auto-organisation spontanées naturelles. Dans ce travail, nous nous intéressons à la synchronisation en opposition de phase, c'est-à-dire une désynchronisation, en utilisant une approche bio-inspirée. Ce type de synchronisation se rencontre notamment chez certaines espèces de grenouilles [Aih09]. En effet, les mâles émettent des croassements périodiques. Pour que les femelles puissent les reconnaître, il ne faut pas de recouvrement entre les croassements de différents mâles. Ces derniers, en décalant progressivement leurs croassement, convergent vers un état où les croassements sont également espacés dans le temps. Dans ce travail, nous évaluons l'intérêt d'appliquer un algorithme similaire pour éviter les collisions de *beacons* dans les réseaux véhiculaires.

2 La désynchronisation bio-inspirée

Dans cette section, nous présentons un modèle mathématique et une description des algorithmes de désynchronisation bio-inspirée considérés.

2.1 Le modèle



(a) À t_1 le noeud 2 envoie son *beacon* et remet sa phase à zéro

(b) Juste après l'emission, le noeud 2 calcule sa nouvelle phase Φ'_i

(c) Les phases des noeuds continuent leurs rotations

FIGURE 1: Phase model

De manière similaire à [Aih09, MSM12], nous utilisons un modèle de phase pour modéliser ces algorithmes. La phase $\Phi_i(t)$ du noeud i est prise par rapport à l'instant d'émission du noeud. Comme représenté sur la Figure 1, la phase évolue sur un cercle linéairement avec le temps. Dans la littérature, différentes définitions de la phase existent, par exemple : $\Phi_i(t) \in [0,1]$ (dans ce cas, la phase est la proportion de cercle parcouru), $\Phi_i(t) \in [0,2\pi]$ (dans ce cas, la phase est un angle par rapport à une ligne verticale passant par le centre du cercle). Dans cet article, nous prenons $\Phi_i(t) \in [0,T]$, $\Phi_i(t)$ est donc une fonction en dent de scie qui représente le délai depuis la dernière émission de *beacon* : $\Phi_i(t) = T(\frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor)$

Un noeud peut mettre à jour sa phase une fois par période, il se produit alors un saut, une discontinuité de phase. Cette mise à jour est faite en fonction des phases des autres noeuds avec lesquels il peut communiquer. C'est elle qui permet la désynchronisation. La phase mise à jour est notée $\Phi'_i(t)$:

$$\Phi'_i(t) = f(\Phi_1, ..., \Phi_N, \boldsymbol{\alpha}) \tag{1}$$

où $\Phi_1, ..., \Phi_N$ sont les phases mesurées des N noeuds du réseau et $\boldsymbol{\alpha}$ un vecteur de paramètre. En pratique, c'est la différence de phase entre le noeud mettant à jour sa phase et les autres qui est souvent utilisée. Un noeud apprend les phases des autres noeuds grâce aux messages qu'il reçoit de ceux-ci. Dans la littérature, la mise à jour de phase peut se produire sur réception d'un message [DRPN07] ou bien juste après l'émission du message du noeud [MSM12], comme représenté par la Figure 1b.

2.2 Les algorithmes de mise à jour considérés

Dans cet article, nous évaluons les performances des algorithmes de désynchronisation dans le contexte de l'ITS. Un algorithme de désynchronisation est défini par la fonction de mise à jour f (équation (1)) et l'instant auquel cette mise à jour est effectuée.

2.2.1 Frog

Nous appelons *Frog*, l'algorithme proposé dans [MSM12] qui est inspiré des croassements de grenouilles. Il est destiné, à la base, aux réseaux de capteurs sans fil. Pour cet algorithme, la mise à jour de la phase s'effectue juste après l'émission du *beacon*. La fonction de mise à jour est la suivante :

$$\Phi_i'(t) = \Phi_i(t) + \frac{1}{\frac{1}{T} + \sum_{j=1}^{N} \alpha e^{-d(\Delta_{ij})} \sin(\Delta_{ij})} \tag{2}$$

avec $\Delta_{ij} = \frac{2\pi(\Phi_i(t)-\Phi_j(t))}{T}$. Étant donné que i vient d'émettre, $\Phi_i(t) = 0$. Les $\Phi_j(t)$ correspondent aux phases des autres noeuds apprises durant la période précédente. α est un paramètre pris dans $[0,1]$ et $d(\Delta_{ij}) = \min(-\Delta_{ij}, 2\pi + \Delta_{ij})$. On note que la mise à jour dépend des phases de tous les noeuds avec plus de poids pour ceux qui ont une phase proche de celle de i (dû au facteur exponentiel).

2.2.2 DESYNC et V-DESYNC

DESYNC [DRPN07] est un algorithme destiné à l'accès au médium dans les réseaux de capteurs sans fil. Pour cet algorithme, la mise à jour s'effectue après l'émission qui suit celle du noeud i. La fonction de mise à jour est la suivante :

$$\Phi_i'(t) = (1-\alpha)\Phi_i(t) + \alpha\Phi_{mid}(t) \tag{3}$$

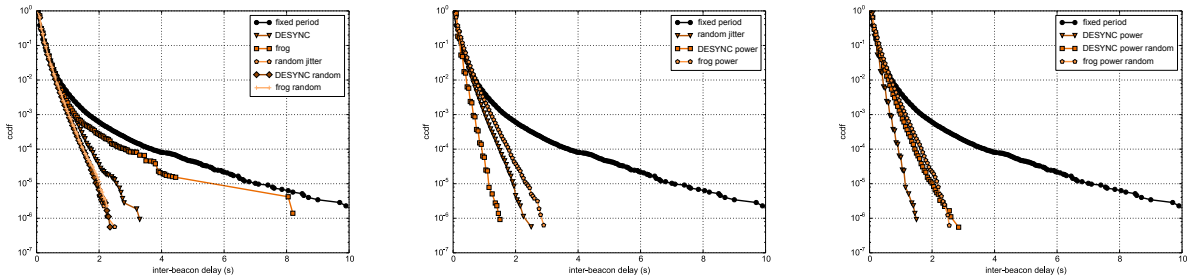
avec $\Phi_{mid}(t) = \Phi_{i-1}(t) + \frac{1}{2}([\Phi_{i+1}(t) - \Phi_i(t)] - [\Phi_i(t) - \Phi_{i-1}(t)])$ le point médian entre les phases des noeuds précédents et suivant (sur le cercle). V-DESYNC [SCIR12] est une variante randomisée de DESYNC proposée pour les réseaux véhiculaires. Une variable aléatoire $X \sim \mathcal{U}(-\frac{a}{2}, \frac{a}{2})$ (a est une fraction de la période) est ajoutée à l'équation (3) pour éviter le problème du terminal caché qui ne permet pas une désynchronisation. Nous verrons dans la section 3 que la randomisation n'est en fait pas la meilleure stratégie pour éviter ce problème.

2.2.3 DESYNC Power

Nous proposons une variante de DESYNC qui consiste à apprendre les phases des noeuds précédents et suivants non seulement par la réception de message, mais par la détection de signaux sur le canal. Cela permet d'effectuer une désynchronisation basée sur le graphe d'interférence et non seulement celui de communication. Ainsi, on évite les collisions non résolues par 802.11p, notamment dues au problème du terminal caché.

3 Simulations et résultats

Les résultats présentés dans cette section sont issus de simulations sous NS2. Le modèle de mobilité provient de la trace de Cologne [UF11] qui est un micro modèle très réaliste. On compare les algorithmes évoqués dans la section précédente à un algorithme d'émission de *beacon* purement périodique ainsi qu'à une version randomisée nommée *Random Jitter* [KHdPMS14]. La Figure 2 représente la fonction de répartition inverse empirique du délai inter-*beacon* mesuré durant les simulations. Ce délai est intéressant, car il correspond au temps passé sans information sur un des voisins du véhicule considéré. Par conséquent, plus la répartition est concentrée proche de 0 plus les informations sont à jour. Sur la Figure 2a, on constate que même si les algorithmes DESYNC et *Frog* sont meilleurs que l'algorithme périodique, ils sont moins bons que *Random Jitter*. On observe également que des versions randomisées de DESYNC et *Frog* ont des performances similaires à *Random Jitter*. Sur la Figure 2b, on compare les versions basées sur la détection de signal sur le canal de DESYNC et *Frog* : *DESYNC Power* et *Frog Power*, avec le meilleur et le moins bon des algorithmes évalués précédemment : *Random Jitter* et l'algorithme périodique. On constate que *DESYNC Power* est meilleur que *Random Jitter*, mais ce n'est pas le cas de *Frog Power*. Enfin, on peut observer



(a) algorithmes classiques et versions randomisées (b) désynchronisation basée sur la détection d'énergie sur le canal (c) désynchronisation basé sur la détection d'énergie sur le canal et randomisation

FIGURE 2: Fonction de répartition inverse du délai inter-beacon

sur la Figure 2c que les version randomisées de *Frog Power* et *DESYNC Power* se comportent moins bien que la version simple de *DESYNC Power*. Nous en tirons la conclusion que l'ajout d'un facteur aléatoire pour éviter les collisions n'est pas bénéfique si l'algorithme est déjà basé sur le graphe d'interférence.

4 Conclusion

Dans cet article, nous avons étudié l'application d'algorithmes de désynchronisation bio-inspirée au *beaconing* dans les réseaux véhiculaires. Nous avons proposé une variante de *DESYNC* : *DESYNC Power* qui est basée sur le graphe d'interférence au lieu de celui de communication. Nous montrons que cette variante fournit des délais inter-*beacon* plus petits que les algorithmes précédents lors de simulations avec un modèle de mobilité réaliste.

Références

[Aih09] Ikkyu Aihara. Modeling synchronized calling behavior of japanese tree frogs. *Physical Review E*, 80(1) :011918, 2009.

[DRPN07] Julius Degesys, Ian Rose, Ankit Patel, and Radhika Nagpal. Desync : self-organizing desynchronization and TDMA on wireless sensor networks. In *Proc. of the 6th international conference on Information processing in sensor networks*, pages 11–20. ACM, 2007.

[HL10] Hannes Hartenstein and Kenneth Laberteaux. *VANET : vehicular applications and internetworking technologies*. Wiley Online Library, 2010.

[KHdPMS14] Bernhard Kloiber, Jerome Harri, Fabian de Ponte Muller, and Stephan Sand. Random transmit jitter against correlated packet collisions in vehicular safety communications. In *6th Int. Symp. on Wireless Vehicular Communications (WiVeC),*, pages 1–5. IEEE, 2014.

[MSM12] Akira Mutazono, Masashi Sugano, and Masayuki Murata. Energy efficient self-organizing control for wireless sensor networks inspired by calling behavior of frogs. *Computer Communications*, 35(6) :661–669, 2012.

[SCB11] Razvan Stanica, Emmanuel Chaput, and Andre-Luc Beylot. Broadcast communication in vehicular ad-hoc network safety applications. In *IEEE CCNC*, Las Vegas, USA, 2011.

[SCIR12] Tossaphol Settawatcharawanit, Supasate Choochaisri, Chalermek Intanagonwiwat, and Kultida Rojviboonchai. V-desync : Desynchronization for beacon broadcasting on vehicular networks. In *IEEE VTC Spring*, pages 1–5, 2012.

[Str03] Steven Strogatz. *Sync : The emerging science of spontaneous order*. Hyperion, 2003.

[UF11] Sandesh Uppoor and Marco Fiore. Large-scale urban vehicular mobility for networking research. In *IEEE VNC*, pages 62–69, Amsterdam, The Netherlands, 2011.