On the Use of Bio-Inspired Desynchronization Algorithms for Beaconing in ITS

Alexandre Mouradian, Véronique Vèque

On the Use of Bio-Inspired Desynchronization Algorithms for Beaconing in ITS

Alexandre Mouradian and 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

***Abstract*—Road safety applications heavily rely on periodic message exchanges known as beaconing in order to gain awareness about the current situation within the vehicle communication range. The high level of transmission contention in dense vehicular networks can induce successive beacon collisions. These collisions impede the vehicles to have an up-to-date view of the environment which can lead to bad decisions from the applications and in the worst case create car crashes. Many propositions of the literature try to tackle this problem by modifying the beacon emission period and/or power either randomly or based on the network density or other measured indicators. In this paper, we explore the possibility to use bio-inspired desynchronization algorithms to avoid beacon collisions. The main idea behind these algorithms is to make nodes converging toward a consensus where they do not emit their beacon at the same time in order to avoid collisions. We show that desynchronization algorithms, if properly implemented, can significantly reduce the probability to have large inter-beacon delays.**

***Keywords*—Anti-phase synchronization, beaconing, distributed algorithm, Performance Evaluation**

I. Introduction

Road safety applications are seen as one of the main incentives to deploy Intelligent Transportation Systems (ITS). Nevertheless, reaching the minimum Quality of Service requirements for the safety applications in dense large-scale mobile wireless networks such as vehicular networks is still an open issue. Road safety applications heavily rely on periodic message exchange known as beaconing in order to gain awareness about the current situation within the vehicle communication range [1]. The high level of transmission contention in dense vehicular networks can induce successive beacon collisions. These collisions impede the vehicles to have an up-to-date view of the environment which can lead to bad decisions from the applications and in the worst case create car crashes. In the current ITS standards [2], the beacons are periodically broadcasted using variants of the IEEE 802.11 standard. Previous works have highlighted the fact that IEEE 802.11 broadcasting is not reliable [3], [4], [5], [6] especially in high density networks such as VANETs because of non-resolved beacon collisions. This leads to unacceptable inter-beacon delays for the target safety applications.

In this paper, we study the application of several bio-inspired beaconing algorithms to reduce beacon collisions while conserving the ITS architecture (notably the 802.11 mechanisms). The idea is to desynchronize the beacons of the vehicles in the same interference domain in order to avoid collisions.

Spontaneous decentralized synchronization (which can also be seen as agreement or consensus) can be observed in many contexts in nature [7]. We can cite the synchronization of fireflies flash in south-Asia, the cooperative motion of bacterias or flocks of birds, and cardiac cells beating at unison, as few of the numerous examples of spontaneous self-organization found in nature. In this work, we are interested in bio-inspired anti-phase synchronization [8].

The contributions of this paper are as follows.

- We compare several desynchronization algorithms in the context of vehicular networks. We also compare these algorithms to the *Fixed Period* beaconing and its randomized version *Random Jitter* [9].
- We propose a modification of a desynchronization algorithm which achieves the best performance of the study.
- We evaluate all these algorithms in a realistic simulation setting using the Cologne mobility trace [10].

The paper is organized as follows. Section II presents and comments the related work. Section III gives the theoretical model, describes the considered algorithms and provides some insight about convergence. In Section IV, we describe the simulation setting and we comment the results. Section V provides conclusions and future works are discussed.

II. Related work

The non-scalability of beaconing using 802.11p has been largely discussed in the literature [3], [4], [5]. It mainly comes from the fixed contention window, the absence of RTS/CTS and the absence of losses detection in 802.11p broadcast mechanism.

As surveyed in [11], there is a variety of solutions to control beacon congestion in vehicular networks. Beacon frequency, emission power, congestion window, coding rate, etc. can be adjusted based on numerous parameters (congestion on the channel, speed of the vehicle, randomly, etc.). In this work, we do not focus on adapting these parameters based on the situation of the vehicle or the channel, but we are instead interested in the category of schemes for which the emission power is fixed and the beaconing frequency can be perturbed provided that its mean converges toward a known value (10Hz

in the simulations). We think that this basic type of schemes may enrich more complex schemes (with period and emission power adaptation) and improve their performance. Moreover, they can be totally implemented at the application (or facilities) layer without requiring any modification of the lower layers (we do not intend to control the contention window or coding rate for instance).

The *Random Jitter* algorithm presented in [9] also falls into this category. It is one of the most simple to implement. The idea is to add a random perturbation to the beaconing period so that successive beacon collisions are made highly improbable. The authors in [9] show that their scheme highly outperforms *Fixed Period* beaconing: the probability to have high inter-beacon delays is reduced. In Section IV, we compare this scheme to several desynchronization algorithms. Up to our knowledge, it is the first work to perform such a comparison.

In the remainder of this section we focus on desynchronization algorithms used to schedule transmissions and/or activity in wireless networks.

The *DESYNC* algorithm [12] have been proposed for single-hop sensor networks in order to interleave periodic reports of the nodes. The principle is quite simple as among a set of n nodes generating reports with a common, fixed period T, the nodes adjust their report date such that they are evenly distributed throughout the time period (they are equally spaced in time with T/n intervals). For that, each node i keeps track of the report times of its two phase neighbors, i.e. the node whose report is just before and the one just after. Using these times, node i can compute the midpoint of its neighbors, and jumps its report date towards it. The details of the algorithm are presented in Section III-B2. The authors show that the system converges to a state in which all nodes are evenly spread out with a spacing of T/n. The algorithm is simple, decentralized, and requires constant memory per node regardless of the network size. Moreover, if nodes are added or removed, the scheme self-adjusts to re-equalize the report intervals. The authors demonstrate the performance of their desynchronization algorithm by applying it to a TDMA MAC protocol which requires no global clock and allows nodes to self-adjust according to the number of participating nodes.

The authors in [13] provide a model of the calling behavior of Japanese male tree frogs, they compare their model to experimental data. The frogs are able to desynchronize their calls so that they do not overlap with the others. The authors propose an anti-phase synchronization algorithm to model this behavior. The phase shift function proposed in [13] is used in the context of wireless sensor networks in [14]. The authors apply the frog model behavior to control the duty cycle of sensor nodes. They desynchronize the activity period of the nodes so that they do not sense and transmit their measure to the base station at the same time.

In [15] the authors propose what appears to be the first application of spontaneous desynchronization to vehicular networks. They propose V-DESYNC which is a slight modification of *DESYNC* [12] in order to : (1) take care of initial beacon collisions (which are not resolved by *DESYNC*) and (2) accept nodes with different emitting (beaconing) periods. They propose to introduce a random perturbation to the phase update of *DESYNC* in order to resolve collisions which can occur in dynamic networks such as VANETs. They show that their modified version of the algorithm performs better than the legacy *DESYNC* and constant period beaconing. Nevertheless, as we show in this paper, it does not outperform the constant periodic beaconing with random jitter proposed in [9]. We thus propose an alternative which is based on channel monitoring at the physical layer (instead of desynchronization based only on received beacons). The remaining collisions resolution being taken care of by the 802.11p layer. This alternative version is shown to outperform both *Random Jitter* beaconing and the randomized version of *DESYNC*.

In the remainder of the paper, we describe the theoretical model and the evaluated protocols before introducing our algorithm, *DESYNC Power*. We then evaluate and compare the performance of the proposed and cited algorithms in a realistic simulation setting.

III. Bio-inspired desynchronization

In this section, we introduce the theoretical model used to design and analyze desynchronization algorithms. We describe the algorithms which are evaluated in Section IV and we provide some insights about their convergence.

A. Theoretical model

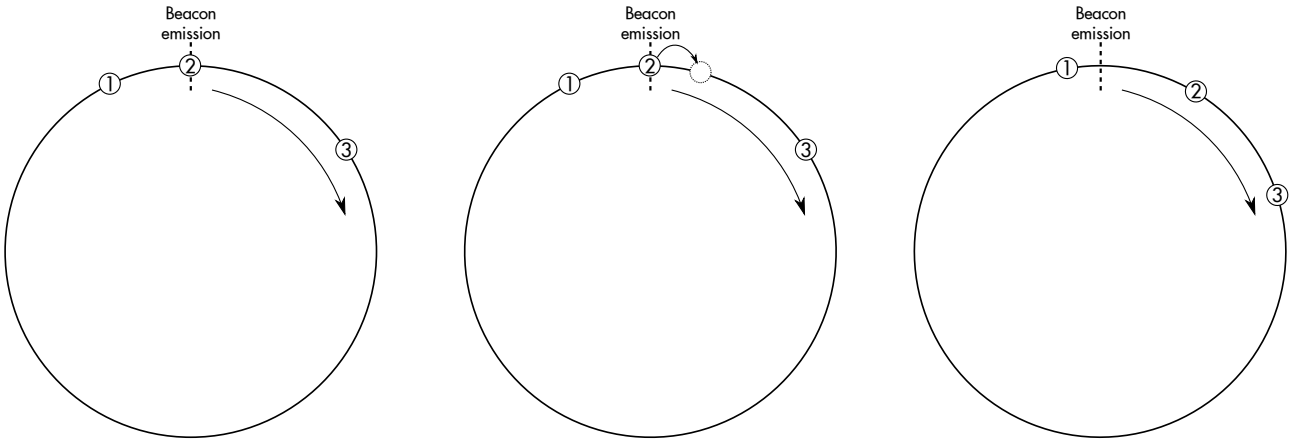
As in most of the previous works, we use a phase model to describe the behavior of the nodes. The phase $\Phi_i(t)$ of each node i is taken with respect to its emission date. The phase evolves on the circle linearly with time, as represented in Fig. 1, and it takes T seconds to complete the revolution. When it reaches the emitting mark (every T seconds), the node sends a beacon and its phase is set to 0 (as depicted for node 2 in Fig. 1a). In the literature, different phase definitions are adopted i.e. $\Phi_i(t) \in [0, 1]$ where the phase represents the proportion of time elapsed for the current revolution, or $\Phi_i(t) \in [0, 2\pi]$ for which the phase is an angle. In this paper, we define $\Phi_i(t) \in [0, T]$. So $\Phi_i(t)$ is a sawtooth function (it represents the delay since the last beacon emission):

$$\Phi_i(t) = T(\frac{t}{T} - \left\lfloor \frac{t}{T} \right\rfloor) \tag{1}$$

A node can update its phase once during each period, and it does so according to the measured phases of the nodes it can communicate with. The updated phase is noted $\Phi'_i(t)$:

$$\Phi'_i(t) = f(\Phi_1, ..., \Phi_N, \boldsymbol{\alpha}) \tag{2}$$

where $\Phi_1, ..., \Phi_N$ are the measured phases of the N nodes of the network and $\boldsymbol{\alpha}$ is a vector of parameters. In practice, the phase differences between the node updating its phase and the other nodes are often considered. A node learns the phases of other nodes through communication (message or signal). In the literature, the update can take place on receiving a message from another node [12] or just after the node message emission [14] as depicted in Fig. 1b.



(a) At t_1 node 2 sends its beacon and resets its phase

(b) Just after the emission, node 2 computes its phase update Φ'_i

(c) The nodes phases continue to rotate

Fig. 1: Phase model

B. Considered update algorithms

In this paper we evaluate the performance of desynchronization algorithms in the context of beaconing for ITS. A desynchronization algorithm is defined by the update function f from equation (2) and the instant at which the function is applied. For the algorithms described in Section II, we give the function f and finally, we propose a new algorithm called *DESYNC Power*.

1) Frog: We call *Frog*, the algorithm proposed in [14] which is inspired by the calling behavior of Japanese tree frogs. For this algorithm, the phase update takes place right after the node beacon emission. The update function is as follows:

$$\Phi'_i(t) = \Phi_i(t) + \frac{1}{\frac{1}{T} + \sum_{j=1}^{N} \alpha e^{-d(\Delta_{ij})} \sin(\Delta_{ij})} \tag{3}$$

with $\Delta_{ij} = \frac{2\pi(\Phi_i(t) - \Phi_j(t))}{T}$. Since node i have just emitted when the phase is updated, we have $\Phi_i(t) = 0$. The $\Phi_j(t)$s correspond to the phases of the other nodes j recorded during the previous period. α is a parameter taken in $[0, 1]$ and $d(\Delta_{ij}) = \min(-\Delta_{ij}, 2\pi + \Delta_{ij})$. We can remark that the phase update depends on the phases of all the other nodes with more influence of the nodes with a closer phase (because of the exponential factor).

2) DESYNC: For this algorithm, the phase update takes place right after the emission of the beacon which follows the considered node i emission. The update function is given by:

$$\Phi'_i(t) = (1 - \alpha)\Phi_i(t) + \alpha\Phi_{mid}(t) \tag{4}$$

where α is a parameter taken in $[0, 1]$ that scales how far node i moves from its current phase towards the desired midpoint, and with $\Phi_{mid}(t)$ the midpoint between phases of the next $(i - 1)$ and previous $(i + 1)$ nodes to emit:

$$\Phi_{mid}(t) = \Phi_{i-1}(t) + \frac{1}{2}([\Phi_{i+1}(t) - \Phi_i(t)] - [\Phi_i(t) - \Phi_{i-1}(t)]) \tag{5}$$

3) V-DESYNC: In this algorithm, a random perturbation is added to the phase update computed by *DESYNC*:

$$\Phi'_i(t) = (1 - \alpha)\Phi_i(t) + \alpha\Phi_{mid}(t) + X \tag{6}$$

with $X \sim \mathcal{U}(-\frac{a}{2}, \frac{a}{2})$ and a, a fraction of the beaconing period. In the remainder of the paper we use 10% of the beaconing period (it is the same value as in [15]). We refer to this algorithm as *DESYNC Random*.

We also propose a randomized version of *Frog* in the same way. We name it *Frog Random*.

4) DESYNC Power: We propose a version of *DESYNC* in which the phases of the next and previous nodes are not measured based on received beacons, but on the power monitored on the radio channel. It allows to perform the desynchronization based on the interference range and not only on the communication range. This avoids the collisions which cannot be resolved by 802.11p in broadcast. Moreover, if the energy detection threshold is lower than the carrier sense threshold, the hidden terminal problem can also be avoided. The only requirement is that the physical layer triggers an interruption whenever the received signal is over a threshold. The application layer should note the instant at which the interruption is received in order to compute the phase of the emitting node. In section IV we show that this proposition outperforms the other desynchronization algorithms.

We also propose to extend *Frog* with this feature and name it *Frog Power*. In the next section, we evaluate randomized versions of these propositions as well.

C. Some considerations about convergence

There are numerous contributions in the literature [16], [17] on the theoretical convergence of consensus algorithms for multi-agent systems with communications (usually, a time varying communication graph is considered). One of the strongest results can be found in [16]. It states that, under the assumption that the update function is convex, if the communication links are bidirectional and during every interval of the form $[t_0, +\infty)$ each agent sends information to each

other through direct or indirect communication, the algorithm converges towards a global consensus. The author also shows that synchronization algorithms fall into this category. In our case, we do not even need a global consensus because beacons of non-interfering nodes can overlap safely.

The authors of *DESYNC* also propose a simple proof of the convergence of their scheme in [12]. The proof shows that the recurrence relation between two successive states of the system converges toward a fixed point. Moreover, the authors conjecture that the convergence speed is in $\mathcal{O}(N^2)$. Nevertheless, the proof is valid only if a node can actually receive packets from the ones which emit just before and after it. This is not always the case in wireless networks. This is the reason why we propose the *DESYNC Power* variation.

In [18], the authors show the convergence of *DESYNC* when the nodes can measure the phases of their neighbors with a Gaussian error. This setting is very similar to the randomized version of *DESYNC*, nevertheless the proof heavily relies on the normality of the perturbation whereas the randomized version of *DESYNC* is uniformly perturbed. Nonetheless, following the result of [16], we can argue that the randomized version of the desynchronization algorithm should converge if the random perturbation keeps the convexity of the update function. In practice, we observe that it is the case in the simulations for the considered randomized algorithms.

IV. Simulations and results

In this section we present the simulation environment and we comment the results.

A. Simulation setup

In this paper, we use the *ns2* [19] simulator with the 802.11p model from [20]. The simulation parameters are detailed in Table I. Concerning the mobility of the vehicles, we used a realistic micro-mobility trace of the city of Cologne [10]. The trace covers a 400 km^2 area during a period of

We consider the algorithms described in the previous section. They only take one parameter α. We choose $\alpha = 0.95$ because it is shown in [15] that a high α (close to one) is beneficial in the context of highly time-varying graphs such as vehicular networks. We compare the desynchronization algorithms to *Fixed Period* beaconing where the node emits a beacon every T seconds and to the *Random Jitter* algorithm for which the period is randomly perturbed. As in [9], we use a uniform perturbation in a $[-\frac{T}{2}, \frac{T}{2}]$ window around the target emission date.

Parameter	Value
Bitrate	6 Mbps
Transmission power	10 dBm
Simulation area	1000x1000 m
Beacon size	400 bytes
MAC and Phy	802.11p
Propagation model	Nakagami m=1
Beacon frequency	10Hz
α	0.95

TABLE I: Simulation parameters.

We are interested in the inter-beacon delay at the receivers because it indicates the amount of time spent without knowledge of a given neighbor. We thus measure, for every pair of nodes, the time between two beacons at the receiver side. This quantity is not equal to the beacon emission period because of packet losses due to random attenuation on the channel (Nakagami model) and collisions. Since the packet losses depend on random parameters, the inter-beacon delay is a random variable. We will thus focus on the distribution of the inter-beacon delay. As in [9], we use the experimental Complementary Cumulative Distribution Function (CCDF) because it allows to compare finely the tails of the distributions. The tails are of interest because they show the probability to observe large delays and thus to miss deadlines. The CCDF for a random variable X is given by $y = P(X > x)$. In the simulation results we only consider the packets received from emitter less than 500 m away from the receivers. Even if some packets are effectively received from farther nodes, we consider they are not relevant for safety applications.

In the next subsection, we present and comment the simulations results.

B. Results

Before commenting the simulations results, we would like to emphasize that the load is very high given the simulation parameters: it takes 1 ms to send a 400 bytes beacon at 6 Mbps with a 1/2 coding rate. Thus, if we have a perfect scheduling, we can accommodate 100 beacons during a period of 100 ms. In the simulation trace we use, there are around 130 vehicles in the 1 square kilometer area and obviously 802.11p does not provide perfect TDMA scheduling.

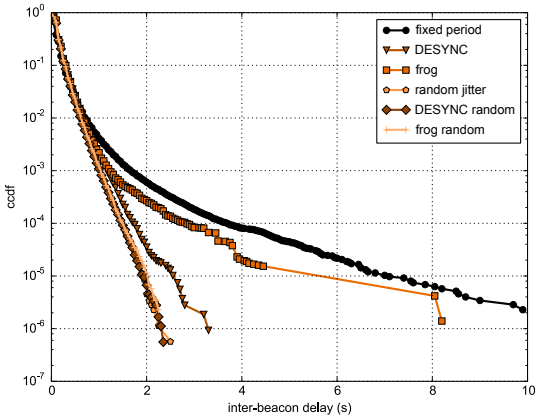


Fig. 2: CCDF of inter-beacon delays: regular algorithms and randomized versions

Fig. 2 depicts the comparison of the experimental CCDF of the *Fixed Period* beaconing, *Frog*, *DESYNC* and their randomized versions. The y axis uses a logarithmic scale. The algorithm with the higher decreasing slope is the best. The *Fixed Period* algorithm is the worst, as expected, because if beacons collide and are not resolved by 802.11p (hidden terminal problem for instance), then the collisions remain throughout the simulation. We observe that *Frog* and *DESYNC* improve the fixed beaconing. *DESYNC* provides a far better improvement than *Frog*. Then, the randomized versions of the algorithms provide the best results. It is interesting to remark

that all the 3 randomized algorithms give almost the same results. This means that the randomization of *DESYNC*, praised in [15], is not better than using a *Random Jitter*; the latter being far easier to implement.

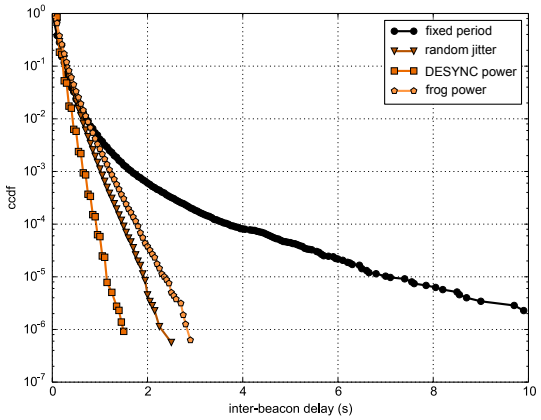


Fig. 3: CCDF of inter-beacon delays: desynchronization based on power detection on the channel

In Fig. 3, we compare the power monitoring versions of *DESYNC* and *Frog* proposed in Section III-B4 to the fixed and *Random Jitter* solutions. We observe that *DESYNC Power* is better than the *Random Jitter* solution, but it is not the case for the power monitoring version of *Frog*. We note that the highest observed inter-beacon delay for *DESYNC Power* is less than 2 seconds whereas it is more than 10 seconds for the *Fixed Period* beaconing.

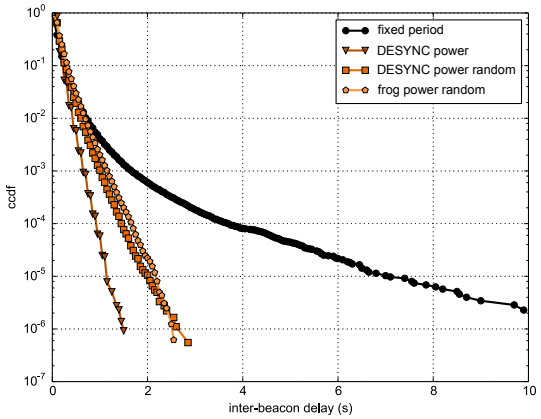


Fig. 4: CCDF of inter-beacon delays: desynchronization based on power detection on the channel and randomization

Fig. 4 depicts the comparison of the *DESYNC Power* with the randomized power monitoring algorithms (we keep the fixed beacon algorithm for reference). The results of the randomized algorithms are actually very close to the ones of the randomized algorithms in Fig. 2. We observe that the non-randomized version of *DESYNC Power* is actually better than the randomized versions of both *Frog power* and *DESYNC Power*. This indicates that, when using channel monitoring, adding a random perturbation actually increases the number of collisions compared to the deterministic algorithm.

The main observations are thus:

- *DESYNC* is better than *Frog* in the context of vehicular networks;
- the randomized version of *DESYNC* does not outperform *Random Jitter*;
- our proposal, *DESYNC Power*, reaches the best performance;
- randomizing *DESYNC Power* actually decreases its performance.

V. Conclusion

In this paper, we studied the application of desynchronization algorithms to beaconing in vehicular networks. The aim is to avoid simultaneous beacon emissions and thus avoid collisions to reduce the inter-beacon delay. We describe a model for the design and study of desynchronization algorithms and we review the existing algorithm of the literature. We propose *DESYNC Power* a variant of the *DESYNC* algorithm in order to take into account the interference graph instead of only the communication graph. We compare the algorithms to *Fixed Period* beaconing and *Random Jitter* beaconing in a realistic simulation setting based on the Cologne mobility trace. We show that the previously proposed desynchronization algorithms could not outperform the *Random Jitter* algorithm, whereas our proposition, *DESYNC Power*, provides shorter inter-beacon delays.

As a future work, we plan to explore the possibility to modify the desynchronization algorithm in order to allow nodes with different beaconing periods and still converge towards a collision free state.

References

[1] H. Hartenstein and K. Laberteaux, *VANET: vehicular applications and inter-networking technologies*. Wiley Online Library, 2010.

[2] ETSI, "102 637-2: Intelligent Transport Systems (ITS)," *Vehicular communications*.

[3] R. Stanica, E. Chaput, and A.-L. Beylot, "Broadcast communication in vehicular ad-hoc network safety applications," in *IEEE CCNC*, Las Vegas, USA, 2011.

[4] H. Noori and B. Olyaei, "A novel study on beaconing for vanet-based vehicle to vehicle communication: Probability of beacon delivery in realistic large-scale urban area using 802.11p," in *IEEE SaCoNeT*, Paris, France, 2013, pp. 1–6.

[5] A. Mouradian, "Study of probabilistic worst case inter-beacon delays under realistic vehicular mobility conditions," in *Ad-hoc, Mobile, and Wireless Networks: 14th International Conference, ADHOC-NOW*, Athens, Greece, 2015, pp. 390–403.

[6] A. Mouradian and V. Vque, "Dissemination strategies in realistic v2v highway networks: The madrid trace case," in *Wireless Days (WD)*, March 2016, pp. 1–3.

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

[8] M.-C. Ho, Y.-C. Hung, and C.-H. Chou, "Phase and anti-phase synchronization of two chaotic systems by using active control," *Physics letters A*, vol. 296, no. 1, pp. 43–48, 2002.

[9] B. Kloiber, J. Harri, F. de Ponte Muller, and S. Sand, "Random transmit jitter against correlated packet collisions in vehicular safety communications," in *6th Int. Symp. on Wireless Vehicular Communications (WiVeC),*. IEEE, 2014, pp. 1–5.

[10] S. Uppoor and M. Fiore, "Large-scale urban vehicular mobility for networking research," in *IEEE VNC*, Amsterdam, The Netherlands, 2011, pp. 62–69.

[11] S. A. A. Shah, E. Ahmed, F. Xia, A. Karim, M. Shiraz, and R. Noor, "Adaptive beaconing approaches for vehicular ad hoc networks: A survey," *arXiv preprint arXiv:1605.07329*, 2016.

[12] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: self-organizing desynchronization and TDMA on wireless sensor networks," in *Proc. of the 6th international conference on Information processing in sensor networks*. ACM, 2007, pp. 11–20.

[13] I. Aihara, "Modeling synchronized calling behavior of japanese tree frogs," *Physical Review E*, vol. 80, no. 1, p. 011918, 2009.

[14] A. Mutazono, M. Sugano, and M. Murata, "Energy efficient self-organizing control for wireless sensor networks inspired by calling behavior of frogs," *Computer Communications*, vol. 35, no. 6, pp. 661–669, 2012.

[15] T. Settawatcharawanit, S. Choochaisri, C. Intanagonwiwat, and K. Rojviboonchai, "V-desync: Desynchronization for beacon broadcasting on vehicular networks," in *Proc. of Vehicular Technology Conference (VTC Spring),*. IEEE, 2012, pp. 1–5.

[16] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on automatic control*, vol. 50, no. 2, pp. 169–182, 2005.

[17] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[18] D. Buranapanichkit, N. Deligiannis, and Y. Andreopoulos, "On the stochastic modeling of desynchronization convergence in wireless sensor networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5045–5049.

[19] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/, [Online; accessed 28-june-2016].

[20] "Overhaul of IEEE 802.11 Modeling and Simulation in NS-2," http://dsn.tm.kit.edu/english/Overhaul_NS-2.php/, [Online; accessed 28-june-2016].