NOLTA, IEICE

Paper

# A mode-lock free decentralized timing synchronization algorithm for intervehicle ad-hoc networks

*Hisa-Aki Tanaka* [1 a)] *and Kenta Shinohara* [* 1]

[1] *Department of Electronic Engineering, The University of Electro-Communications (UEC)*
*1-5-1 Choufugaoka, Choufu-shi, Tokyo 182-8585, Japan*
[*] *The author is currently with NTT Network Service Systems Laboratories, Japan*

[a)] *htanaka@uec.ac.jp*

Received September 16, 2014; Revised December 5, 2014; Published April 5, 2015

**Abstract:** Timing synchronization is an important integrating component in wireless distributed systems, such as mobile ad-hoc networks, M2M networks, and wireless sensor networks, and therefore, various timing synchronization algorithms have been proposed so far. Recently, Imai and Suzuki developed a new synchronization algorithm based on a time division multiple access (TDMA) protocol. Despite of its efficiency in synchronization for vehicle-to-vehicle communications, their algorithm sometimes suffers from a certain undesired synchronous pattern, *i.e.,* a so called mode-lock state or a deadlock. Although their algorithm takes certain precautions to avoid this mode-lock state, in around more than 10% of instances this state is observed to persist and the desired perfect synchronization is not realized. Then, first we investigate the mechanism of this persisting mode-lock state for their algorithm. With this insight to the mode-lock state, we propose a new mode-lock free (*i.e.,* mode-lock eliminating) distributed algorithm that always leads to a perfect synchronization. From systematic, comparative simulations, we observe that the proposed algorithm always eliminates mode-lock states, and eventually leads to the perfect synchronization. In addition, we observe the algorithm realizes even faster synchronization, compared with the algorithm by Imai and Suzuki, although these observed properties are not mathematically proved in this study.

**Key Words:** timing synchronization, wireless ad-hoc networks, wireless sensor networks, deadlock, mode-lock state, TDMA

## 1. Introduction

Applications of mobile ad-hoc networks have been actively developed for vehicle-to-vehicle communications recently. Such applications are expected to support a safe driving and prevent traffic accidents, and the Japanese ministry of land, infrastructure, transport, and tourism has organized a project for advanced safety vehicles (ASV) [1]. While in Toyota central R&D laboratories, basic researches have



Nonlinear Theory and Its Applications, IEICE, vol. 6, no. 2, pp. 285–294  DOI: 10.1588/nolta.6.285

been carried out for intervehicle ad-hoc networks, which cover a new timing synchronization [2], MAC protocol [3], realistic urban traffic simulations [4], and even field trials [5]. In [2], Imai and Suzuki developed an efficient synchronization algorithm for vehicle-to-vehicle (V2V) communications based on the decentralized time division multiple access (D-TDMA) protocol [3]. The algorithm by Imai and Suzuki includes new devices for synchronizing two (or even multiple) groups of vehicles, and its performance is analyzed under certain realistic environments by systematic simulations so far [4]. However, as they have already pointed out in [2], their algorithm often suffers from a certain undesired synchronous pattern; a so called mode-lock state [6] or a deadlock. Although their algorithm takes certain precautions to avoid this mode-lock state, in more than 10% of instances this state is observed to persist and the desired perfect synchronization is not obtained.

To realize the perfect synchronization, in this study, first we clarify how and why this mode-lock state persists when their algorithm is used. Based on this insight, we propose a new mode-lock free distributed synchronization algorithm which eventually leads to the perfect synchronization. From systematic, comparative simulations, we verify the proposed algorithm always eliminates mode-lock states, and we observe it realizes even faster synchronization, compared with the algorithm by Imai and Suzuki, although these properties are not yet mathematically proved in this study. The basic idea of this study was presented in NOLTA'08 [7]. However, the conference paper lacks a detailed analysis and systematic simulation results of the proposed algorithm. In the present paper we not only provide more detailed analysis but also show systematic simulation results, both of which validate the usefulness of proposed algorithm.

This paper is organized as follows. In Section 2, we explain the background of this study and introduce the distributed timing synchronization algorithm proposed by Imai and Suzuki. In Section 3, we consider and clarify how and why the mode-lock state persists in their algorithm. Then, we propose a new mode-lock free, synchronization algorithm. In Section 4, we compare the performance of the algorithm by Imai and Suzuki and our new algorithm, by systematic simulations. Finally, discussions and conclusions are given in Section 5.

## 2. Background of this study; synchronization for D-TDMA protocol

First, we introduce the D-TDMA protocol [3] as a basis of this study, which is outlined as follows. As shown in Fig. 1, each node (*i.e.,* node A, B, C and D in Fig. 1) is preassigned a single time interval for packet transmission (*i.e.,* a slot with an equal length in Fig. 1), and these slots are evenly spaced along the time in one frame as shown in Fig. 1, in order to avoid packet collisions. The number of slots is generally set to several hundreds, which is much larger than the number of vehicles directly communicating with each other. This point is explained in Section 4, in detail. (Note that the number of slots in Fig. 1 is impractically small, which is just for simplicity of illustration.) Thus, all neighboring nodes (*i.e.,* nodes A, B, C, and D in Fig. 1) send their packets, one after another in one frame, and this sequence is repeated if the nodes are maintained in the group. Since the frame length is equal in all nodes and frames are repeated regularly in each node, we can naturally introduce the phase variable $\phi \in [0, 2\pi]$ as shown in Fig. 1, in which one frame length becomes $2\pi$. Namely, we identify the time $t$ to the phase $\phi$ by $\phi \equiv (t \bmod T) \times 2\pi$, where $T$ is one frame length. This phase variable is useful for understanding the mode-lock state as shown in Section 3. Here we note one important requirement in this TDMA protocol, *i.e.,* a perfect synchronization of local timings (*i.e.,* timers) in neighboring nodes, has to be satisfied in a precision within the guard interval in Fig. 1, otherwise packet collisions may happen. Note that this is the frame synchronization among neighboring nodes, and if this frame synchronization is attained, slots for packet transmission are regularly coordinated among neighboring nodes.

Second, we consider issues of local time synchronization. If we assume a certain global timing is somehow available (for instance, from GPS [8, 9]), each node acquire the global timing independently, and the synchronization can be realized relatively easily. However, in real environments this global timing is not always accessible and hence neighboring nodes have to communicate with each other to realize a common local timing, *i.e.,* a distributed timing synchronization. For distributed timing

synchronization methods in vehicle-to-vehicle communications, there have been a few studies reported so far [2, 10, 11]. These synchronization methods share the same idea in using an 'averaged timing' over neighboring mobile nodes directly communicating with each other. This averaged timing is defined by the average over the local timings of those mobile nodes, which enables each mobile node update (*i.e.*, synchronize) its local timing by referencing the average timing in a fully distributed way. For instance in [2] this synchronization process is repeated in each frame as follows.
(1) Firstly, each node periodically receives the local timing of the $i$-th neighboring nodes ($\equiv t_i$) from preambles in their packets.
(2) Then, secondly, each node periodically obtains all timing errors ($\equiv \Delta t_i = t_i - t_0$) to the $i$-th neighboring nodes' local timings by subtracting its local timing ($\equiv t_0$) from neighboring nodes' timings. ($= t_i$)
(3) Finally, each node updates its local timing such that the average of all timing errors becomes minimized, *i.e.*, the local timing is set to $t_0 + \tau$ with $\tau \equiv n^{-1} \sum_{i=1}^{n} \Delta t_i$, where $n$ is the number of neighboring nodes.

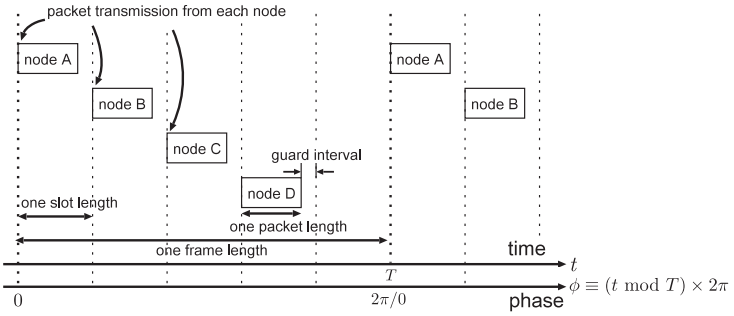


**Fig. 1.** Time structure in the TDMA protocol in this study.

With this synchronization algorithm using averaged timings, groups of vehicles realize a temporally and spatially smooth synchronization of their local timings with each other, and therefore this method is expected to be suitable for the D-TDMA protocol in intervehicle communications [3, 4]. Note that other synchronization methods including the IEEE 802.11 TSF (timing synchronization function; for instance, see [8, 9]) often show an abrupt change of local timings which violates the above 'smooth' synchronization. We also note that distributed timing synchronization methods in this study are natural and reasonable in a sense that the local timing informations can be easily shared through packet communications among neighboring nodes; packets include the local time information in their preambles.

The synchronization algorithm by Imai and Suzuki [2] shares the same idea to the conventional synchronization methods [10, 11], since it is based on the averaged timing. However, compared with these conventional methods the algorithm by Imai and Suzuki [2, 4] includes two new additional devices as below, and hence it can be advantageous over previous methods when applied to larger, distributed intervehicle communication networks. The new devices are as follows.
(i) This algorithm takes into account of a relative degree of synchronization between two (or more) groups of vehicles, and tries to accelerate synchronization over two groups of vehicles [4], and
(ii) this algorithm tries to avoid a certain undesired asynchronous state; a so called mode-lock state or a deadlock, knowing a degree of synchronization for each node to the neighboring nodes by computing the variance of all timing errors in the above (2) [2].
Note that the device (i) is not essentially involved in the situation of this study as well as in [2], because this device is effective only when more than two independent groups of vehicles are going to be synchronized, in which each group of vehicles are already perfectly synchronized. While in this study, neighboring two nodes are generally not perfectly synchronized. We also note that even if this (i) is included in the simulations, we have obtained the same results. Hence, in contrast to (i), the device (ii) becomes essential in this study, and we focus this (ii) and consider its effect in detail, in Section 3.

## 3. A mode-lock eliminating timing synchronization algorithm

As observed in [2], undesired synchronous states; mode-lock states often emerge and persist if no prevention is given in a decentralized synchronization algorithm based on the averaged timing. Although mode-lock states have some variations in their dynamics and complexity ([6, 13, 14]) the basic structure is given as in Fig. 2; all nodes are placed on a two dimensional regular array (just for simplicity) and the central zone of the mode-lock state (*i.e.*, a so called 'core') is set to the center of the array (also just for simplicity). Then, the precise definition of mode-lock states in this particular setting is given as follows. Using the phase variable $\phi$ introduced in Section 2 (see Fig. 1), the local timings of nodes are characterized as in Fig. 2; thicker marked nodes ($\bullet$) correspond to $\phi \simeq 2\pi$. In contrast, thinner marked nodes ($\circ$) correspond to $\phi \simeq 0$. Nodes with the same phase symmetrically radiate from the central zone of the mode-lock state. Hereafter, we call central zone of the mode-lock as core, for simplicity. Around the core, the phases of nodes are evenly ordered, which form a two dimensional rotating wave as $\phi(t)$ increases as time. (A particular example in Fig. 2 exhibits a counter-clockwise rotating wave. The direction of rotation is determined by the choice of initial local timings (*i.e.*, phases $\phi(0)$) of nodes). Note that such a mode-locked state emerges in non-regular networks (including an example shown later in Fig. 4(b)), and even multiple cores can coexist in one mode-locked state [6, 14]. We also note that this rotating, mode-lock state is notorious for its difficulty in elimination; examples are known in VLSI clocking [13], millimeter wave generation [6], and heart diseases [14].

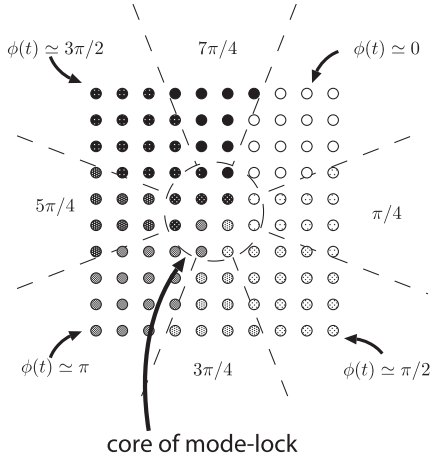


**Fig. 2.** Basic structure of mode-lock state.

### 3.1 Imai and Suzuki's algorithm

The D-TDMA protocol introduced in Section 2 requires one strict constraint; each node is always synchronized with neighboring nodes in the precision within one guard interval. However, in the mode-lock state as shown in Fig. 2, this requirement is clearly violated because nodes around the core have a widely distributed local timings $\phi(t) \in [0, 2\pi]$. (Note that this is nothing to do with evenly spaced packet transmission timings in Fig. 1). Imai and Suzuki observed that this undesired state persists and they include the following precautions for it in their algorithm [2];
(i) once in the frame period $T$, each node judges if it is in the core of the mode-lock state by checking the variance of timing errors to neighboring nodes $n^{-1}\sum_{i=1}^{n}\Delta t_i^2$ (cf. the device (ii) in Section 2); if the variance exceeds a certain threshold, the node assumes it belongs to the core of the mode-lock. Otherwise, the node assumes it is out of the core, and then
(ii) if a node is assumed to be in the core, this node immediately updates its local timing ($\equiv t_0$) to the most recently received local timing ($\equiv t_*$) from neighbouring nodes (namely, $t_0$ is set to $t_*$, instead of updating its timing to the averaged timing given by (3) in Section 2), and this node transmits its updated local timing information to neighboring nodes once in the frame period.

At first glance, these precautions (i) and (ii) seem effective to perfectly synchronize the nodes in the core, and eventually lead to a perfect synchronization over all nodes. However, in reality, what

is often observed (through numerical simulations in Section 4) under these precautions (i) and (ii), is shown as in Fig. 3(a). Here we arrive at the following insight. As indicated by the bidirectional arrows ($\leftrightarrow$), the local time information flow is always bidirectional, which imply nodes in the core, and other (majority of) nodes are coupled with each other. However, majority of nodes outside the core are not so much affected by nodes in the core for the following reasons; (a) the mode-lock state itself is a quite robust pattern, (b) the range of interaction between two nodes is relatively short due to V2V communications in a wide area, and hence, (c) majority of nodes far from the core virtually show no change in their local timings.

Therefore, as shown in snapshots 1, 2, and 3 of Fig. 3(a), the position of the core is gradually shifted due to 'random perturbations' from adjacent nodes to the core; the core keeps slowly meandering. If the core fortunately reaches to the outside boundary of the network, it disappears and this leads to the perfect synchronization. However, as already observed in [2], there is no guarantee for this lucky event within a time of hundreds flames.

### 3.2 Our proposed algorithm

In contrast to such a random perturbing approach, here we propose a simple, but quite effective synchronization algorithm that somehow destroys the mode-lock state itself. The proposed algorithm is basically on the same line of previous methods [2, 10, 11], in a sense that it is based on the averaged timing from neighbouring nodes. However, as opposed to these previous methods, this method destroys the structure of the mode-lock itself in a distributed manner as follows;

(i)$'$ each node judges if it is in the core of the mode-lock state; this step is exactly the same to the above (i). However, the following step is quite different from the above (ii),

(ii)$'$ if a node is aware of being inside or near the core (by using (ii) in Section 2), this node temporally quits sending its (erroneous) local timing information, only receives the local timing information (*i.e.*, $t_i$ in Section 2) from other nodes as indicated by the unidirectional arrow ($\rightarrow$) in Fig. 3(b), and it updates its local timing to the averaged timing. (Note, if such a node does not receive the local timing information from the outside of the core, its local timing becomes quite different to the one in the outside of the core, resulting in violation of 'smooth' synchronization assumed in Section 2.)

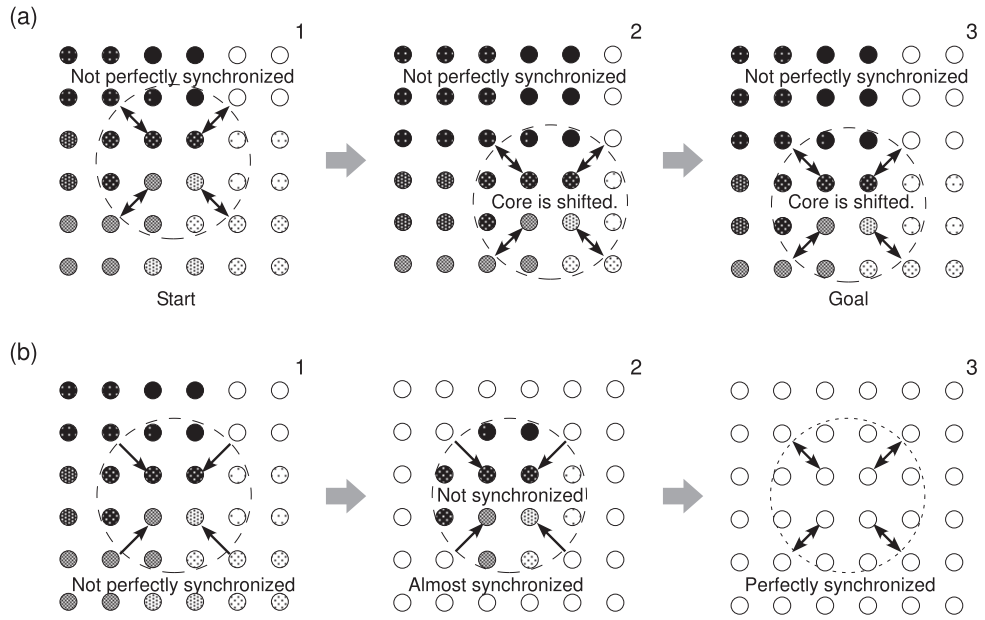


**Fig. 3.** Essential mechanisms of the both two algorithms. (a) Imai and Suzuki's algorithm [2]. (b) Proposed algorithm. Arrows ($\leftrightarrow, \rightarrow$) indicate the flow of local time information.

The algorithm is thus, quite simple in itself. However, when all nodes start synchronizing with each other by this algorithm, the effect appears drastically; a typical process to the perfect synchronization is illustrated as in Fig. 3(b). As indicated by the unidirectional arrows ($\rightarrow$) in snapshots 1 and 2

in Fig. 3(b), the time information flow initially becomes inward to the core. This implies that the adjacent nodes placed outside the core no more receive the time information form nodes in the core, which results in a slight change of the averaged timing at each adjacent node (to the core). This slight change is due to the randomness of the preassigned (*i.e.*, randomly coordinated among the nodes [3]) slots, and this gradually breaks the symmetry of the phases around the core (cf. Fig. 5(b)). This break of symmetry eventually lead to a near synchronization among nodes outside the core (cf. snapshot 2 in Fig. 3(b) and Fig. 5(d)), because perfect synchrony is now the other unique stable state. Once these outside nodes are mutually synchronized, the nodes in the core are synchronized to them, resulting in the perfect synchronization over all nodes (cf. snapshot 3 in Fig. 3(b) and Fig. 5(e)). Note that after snapshot 2 in Fig. 3(b) the core is no more detected (by the above (i)′) and the time information flow becomes bidirectional as in snapshot 3 in Fig. 3(b). A detailed analysis of the above observation is given in Section 4.1, and a systematic performance evaluation for both algorithms is given in Section 4.2.

## 4. Performance analysis for both algorithms

To analyze and evaluate the performance of two algorithms in Section 3, we carry out systematic simulations as follows. Firstly, the simulation setup here faithfully follows the setup in [2], which is summarized as follows.

(i) Each node periodically sends the local timing message to neighbouring nodes in the beginning of its assigned slot (Fig. 1). In this simulation, one frame length is set to 0.1 [s] and the number of slots is set to 400, according to the specification of ASV [1].

(ii) Each node has a given, small frequency mismatch in its timing clock, which ranges in $\pm 0.01\%$ accuracy, according to the specification of the clock oscillator used for ASV [15].

(iii) Each node is uniformly placed on a regular two-dimensional lattice (*i.e.*, array), and each node can communicate with neighbouring nodes in (1) the two-hop distance (*i.e.*, 12 nodes in this radius) or (2) a bit larger distance (*i.e.*, 20 nodes in this radius), as shown in Fig. 4(a) [1]. In this simulation, we assume all events take place in a relatively short time scale (up to several hundreds of flames = a few minutes) in widely distributed, urban traffic environments. Therefore, mobility of each node can be practically ignored here.

(iv) According to the setup in [2], the total number of nodes is set to 1024, and each node with a certain preassigned slot is introduced on the two-dimensional lattice, one by one in the initial 100-frame time at the beginning of the simulation.

(v) As a first approximation, a collision between timing messages is safely ignored. The reason for this approximation is explained as follows. Firstly, a chance of collision is quite rare, since the number of slots is much larger than the number of vehicles directly communicating with each other, and the preassigned slots are coordinated to avoid collisions, which is the advantage of D-TDMA over conventional CSMA/CA (cf. Fig. 18 and Fig. 19 in [4]). Secondly, even if a collision of packets emerges, the timing information is located at the beginning of the packet; the timing information can be safely received although the long data part is lost due to collisions.

### 4.1 Eliminating process of mode-lock states

Next, to correctly visualize the synchronization process and evaluate the synchronizability, we introduce a 'phase coordinate' as in Fig. 5. On this phase coordinate, the local timing of each node (in Fig. 1) is identified to a certain point on the $[0, 2\pi]$ interval. Thus, synchronization process for a given algorithm in this study is clearly visualized as the dynamics of 'phase points' on the phase coordinate, and hence an easy comparison of the mechanisms among different synchronization algorithms becomes possible.

A typical process of mode-lock elimination by the proposed algorithm in Section 3 is visualized as in Fig. 5. Figure 5(a) shows the initial mode-lock state, where seven nodes ($\circ$) in the core are detected by

[1]The same simulations in Section 4 are also carried out for non-regular distribution of vehicles as shown in Fig. 4(b), which result in a similar eliminating process of mode-lock states and synchronizability, respectively shown in Section 4.1 and Section 4.2. Hence, the assumption of 'uniform' distribution (*i.e.*, (iii) in Section 4) is not essential.

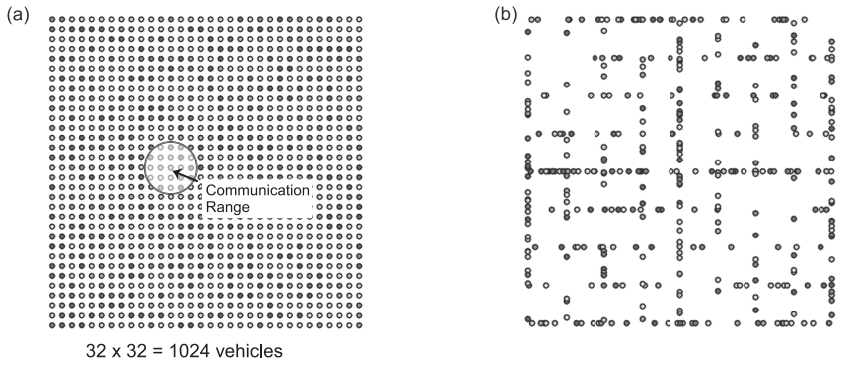


**Fig. 4.** Simulation setup. (a) Case of regular distribution of vehicles. (b) Case of non-regular distribution of vehicles showing the mode-lock state.

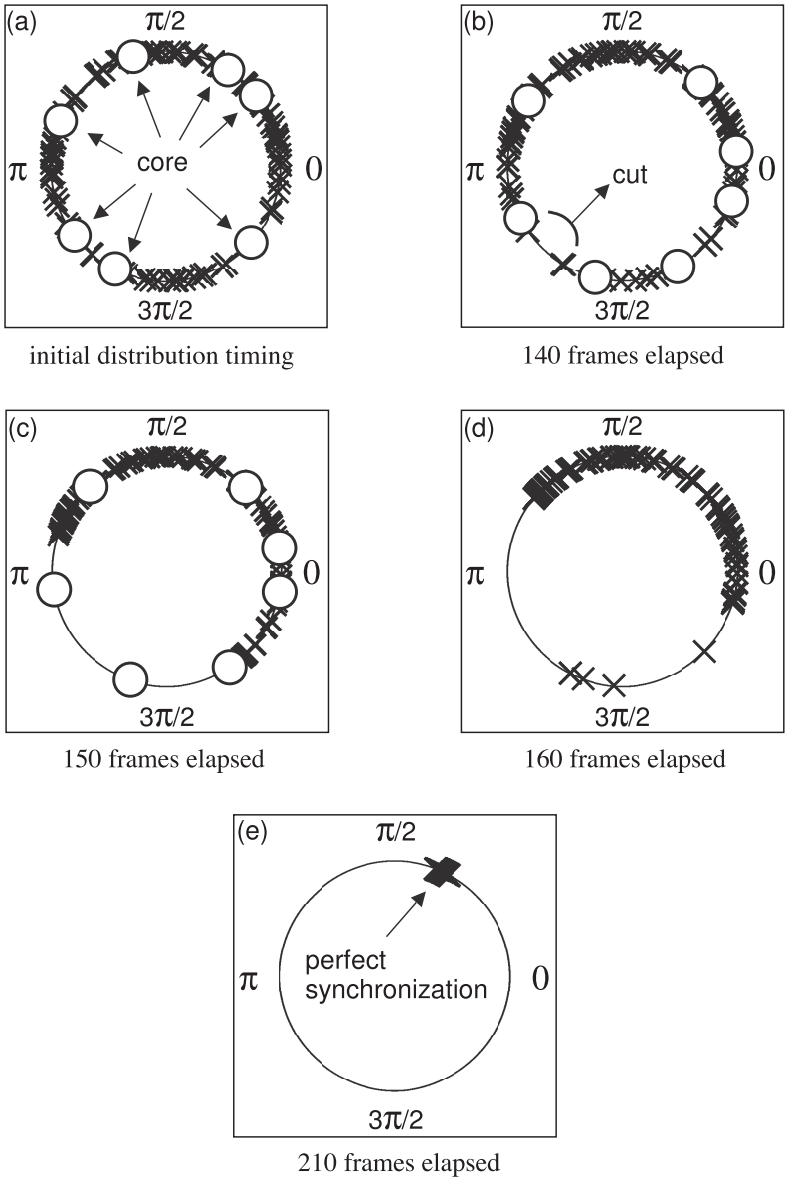


**Fig. 5.** Eliminating process of mode-lock state.

themselves from the algorithm (*i.e.*, (i)′ in Section 3) and other nodes (×) exhibit uniformly distributed timings, which implies a stable rotating wave emerges around the core. However, as shown in Fig. 5(b) a 'cut' (*i.e.*, a void on the $[0, 2\pi]$ interval) appears from the initial uniformly distributed timings (in Fig. 5(a)) as this synchronization algorithm operates. And once this cut appears, it grows rapidly

(Fig. 5(c) and 5(d)), and finally the mode-lock is totally eliminated and the perfect synchronization is achieved (Fig. 5(e)).

As opposed to the proposed algorithm, the original synchronization algorithm by Imai and Suzuki does not have any such eliminating effect to the mode-lock state. Instead, their method introduces a kind of randomly perturbing effect on the position of the core. Therefore, this does not guarantee the elimination of mode-lock in a practical time scale, as mentioned in Section 3.

### 4.2 Comparison of synchronizability for both algorithms

Finally, we provide a systematic performance evaluation for both algorithms. To precisely measure the degree of synchronization among all nodes, the following order parameter $\sigma$ [6] is convenient, which is defined as

$$\sigma(t) = \frac{1}{N}\left|\sum_{i=1}^{N} e^{j\phi_i(t)}\right|, \tag{1}$$

where $\phi_i(t)$ represents the phase of the $i$-th node at time $t$ and $N$ is the total number of nodes. By this definition, $\sigma$ becomes 1 when all nodes are perfectly synchronized, and becomes 0 when their phases (*i.e.*, local timings) are uniformly distributed over $[0, 2\pi]$. It is noted that if the mode-lock persists $\sigma$ becomes less than 1.

Following the simulation setup (i)–(v) in Section 4, systematic simulations are carried out to evaluate the performance for both algorithms, and using the above $\sigma$, 1,000 random trials for both methods for cases of two-hop transmission range and a bit larger transmission range, are shown in Figs. 6(b) and (a) respectively. In each trial, simulation is carried out for 500-frame periods, and the histogram of the final degree of synchronization $\sigma$ is made from 1,000 trials. In addition to the results for Imai and Suzuki's algorithm and our algorithm, the result from a 'simple average' algorithm is also obtained in this simulation. This simple average algorithm means a synchronization algorithm based on the averaged timing without any precautions to mode-lock. Therefore, compared with this simple average algorithm, synchronizability becomes much improved both for the two algorithms (*i.e.*, Imai and Suzuki's and our algorithm), as observed in the both insets in Figs. 6(a) and (b) respectively.

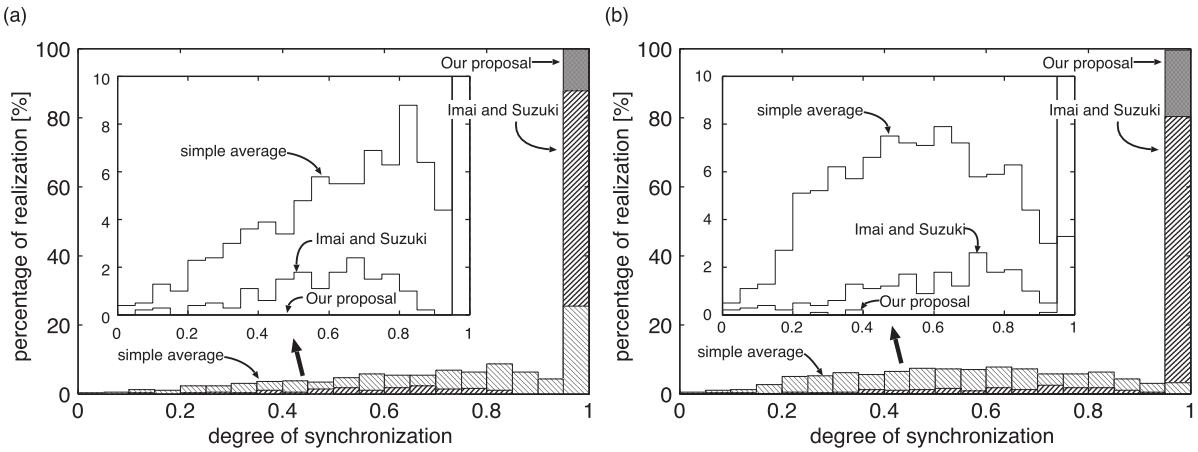


**Fig. 6.** Comparison of synchronizability. The horizontal axis and the vertical axis respectively show the degree of synchronization $\sigma$ and the corresponding percentage of realization. Each inset shows a magnification of the lower part of the histogram, respectively. (a) Case of the two-hop transmission range (*i.e.*, 12 nodes in this radius). (b) Case of a larger transmission range (*i.e.*, 20 nodes in this radius).

As shown in Fig. 6(a), the proposed algorithm realizes a 100% synchronization for 1,000 trials, while Imai and Suzuki's algorithm realizes less than 80% synchronization; more than 20% instances remains unsynchronized and this shows the mode-lock is not yet eliminated for these instances. Also, in Fig. 6(b), near 100% synchronization is realized by the presented method. Here we note that, in these three instances out of 1,000 trials, the mode-lock state is not completely eliminated at the

500-frame periods. However, we have verified these mode-lock states are eliminated by simply taking more time *i.e.*, 1000-frame periods.

## 5. Conclusions

We have presented a simple and effective decentralized synchronization algorithm for intervehicle ad-hoc networks. This algorithm is based on the insight to the nonlinear nature of mode-lock states, and its effectiveness is verified through a series of comparative performance evaluations to the previous synchronization algorithm.

We expect faster and more reliable decentralized synchronization algorithms are obtained by combining the proposed algorithm to the utilization of a certain global timing information in the mobile environments; (i) an algorithm with an efficient utilization of a reliable time information, for instance, from GPS, and (ii) a frequency synchronization algorithm for multiple groups of vehicles with frequent unions or separations with each other. These issues will be reported elsewhere.

## Acknowledgments

We appreciate Dr. J. Imai (Toyota central R&D labs.) for his informative comments. We also appreciate kind helps from Prof. N. Wakamiya (Osaka Univ.) for advanced simulation tools.

## References

[1] Ministry of land, infrastructure, transport, and tourism, "Advanced safety vehicle," http://www.mlit.go.jp/jidosha/anzen/01asv/index.html

[2] J. Imai and N. Suzuki, "Study on a self-adaptive timing synchronization: a method to avoid local optimizations," *IEICE Technical Report*, vol. 107, no. 53, pp. 67–71, 2007.

[3] S. Makido, N. Suzuki, T. Harada, and J. Muramatsu, "Decentralized TDMA protocol for real-time vehicle-to-vehicle communications," *Transactions of Information Processing Society of Japan*, vol. 48, no. 7, pp. 2257–2266, 2007.

[4] Y. Tadokoro, S. Makido, K. Ito, and N. Suzuki, "A proposal of transmission period control scheme for decentralized TDMA protocol in safety applications using inter-vehicle communications," *IPSJ Journal*, vol. 51, no. 3, pp. 945–950, 2010.

[5] S. Makido, H. Hayashi, J. Imai, T. Harada, K. Ito, Y. Tadokoro, H. Tanaka, N. Suzuki, and E. Teramoto, "Reliable MAC protocols for next generation V2V communications: field operation test under simulated high traffic environment," *IEICE Technical Report*, vol. 109, no. 440, pp. 89–94, 2010.

[6] H. Tanaka and A. Hasegawa, "Modelock avoiding synchronization method," *IET Electronics Letters*, vol. 38, no. 4, pp. 186–187, 2002.

[7] K. Shinohara and H. Tanaka, "Mode-lock eliminating timing synchronization algorithm for intervehicle ad-hoc networks," *Proc. NOLTA'08*, pp. 720–723, 2008.

[8] T. Kawazoe, M. Nishijima, K. Sorita, N. Tsuda, and H. Kaneko, "The technical trend of recent car navigation and GPS receiver," *JRC Review*, no. 47, pp. 26–30, 2005.

[9] H. Kondo, "Time and timing generation using GPS receiver and its applications," *ISCIE Journal 'Systems, Control and Information',* vol. 51, no. 6, pp. 273–278, 2007.

[10] Y. Akaiwa, H. Andoh, and T. Kohama, "Autonomous decentralized inter-base-station synchronization for TDMA microcellular systems," *Vehicular Technology Conference, 1991. Gateway to the Future Technology in Motion., 41st IEEE*, pp. 257–262, 1991.

[11] E. Sourour and M. Nakagawa, "Mutual decentralized synchronization for intervehicle communications," *IEEE Trans. Vehicular Technology*, vol. 48, no. 6, pp. 2015–2027, 1999.

[12] H. Tanaka, K. Shimizu, O. Masugata, and T. Endo, "Flexible phase synchronization control method using partially unlocking oscillator arrays," *IET Electronics Letters*, vol. 43, no. 12, pp. 672–674, 2007.

[13] V. Gutnik and A.P. Chandrakasan, "Active GHz clock network using distributed PLLs," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1553–1560, 2000.

[14] N. Bursac, F. Aguel, and L. Tung, "Multiarm spirals in a two-dimensional cardiac substrate," *Proc. National Academy of Science*, vol. 101, no. 43, pp. 15530–15534, 2004.
[15] EPSON, "SG-210S*BA (crystal oscillator output: CMOS) for automotive," http://www5.epsondevice.com/en/quartz/product/osc/spxo/sg210sxba.html