Pulse Synchronization for Vehicular Networks

Cheng-Yu Han, Thomas Nowak, Alain Lambert

Pulse Synchronization for Vehicular Networks

Cheng-Yu Han[1], Thomas Nowak[1] and Alain Lambert[1]

***Abstract*— This paper improves pulse-coupled synchronization for highly dynamic wireless networks, where vehicles may move unpredictably, causing topological changes of the network. For pulse-coupled methods, vehicles broadcast zero-bit pulses to estimate the clock differences to their neighbors. Vehicles then update local clocks according to the average of difference of pulses from its neighbors. The proposed algorithm further introduces (1) a time wheel to improve the robustness of the synchronization protocol and (2) an additional drift compensation mechanism to reduce clock skew. We compare our new algorithm to previous works via simulation. Both static and dynamic networks are simulated and compared. Different frequencies and clock drifts are analyzed. The proposed algorithm adapts to highly dynamic vehicle networks more quickly and more robustly than previous algorithms.**

I. INTRODUCTION

Clock synchronization is a critical issue in vehicular networks [1]–[4]. In particular, it is a cornerstone of sensor fusion, where algorithm work better when data are precisely synchronized rather than time stamped. Distributed data fusion of non-synchronized data necessitates to take into account the shift between the times of the measurements. Such shifts increase the data fusion imprecision [5]–[7]. A quite large number of clock synchronization protocols, mostly for wired networks, have been developed over the past few decades. However, most of them are unsuitable for wireless vehicle networks due to their dynamicity, low bandwidth, and often low-powered computing nodes [8], [9].

In a vehicle network, sensors are not necessarily connected to vehicles. There are supplementary sensors which work independently to help collecting and broadcasting useful information [10]. For those sensors, limits on bandwidth directly influence message exchanges among vehicles [11]. Also, the hardware of sensors is usually very restricted (limited storage, slow computation) because the size of the sensors is small [5], [12]. In addition, unlike wired networks, wireless vehicle networks are dynamic. The vehicles move and change the communication links. In view of this, traditional clock synchronization methods for the wired networks are often not applicable for wireless vehicle networks.

Most clock synchronization algorithms for wireless sensor networks are packet-based methods, where vehicles exchange digital data to agree on a common time base. Centralized approaches [15]–[17] distribute the clock of a master node to other nodes and are not robust to any failure of the master node. In the light of this, an average consensus-based protocol is proposed in [18] to consensus both clock offsets and frequencies. A maximum consensus-based protocol is proposed in [19] to improve the convergence rate. A second-order consensus method is proposed in [20] to further reduce the clock skew.

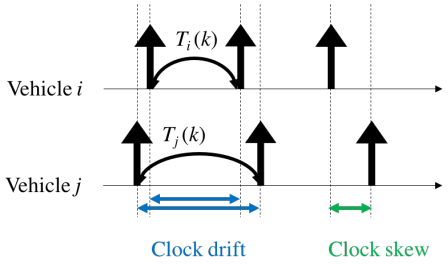


Fig. 1: Virtual clocks (vertical arrows) of vehicle i and j, where the clock drift represents the difference of clock periods between vehicles. The clock skew represents the largest difference of time between sensor in round k.

For all the above packet synchronization [18]–[20], nodes exchange data information (global time, frequency, or system parameters) between its neighboring nodes and agree on some parameters of the model to have a common notion of time. In large vehicle networks, message delivery might be unreliable due to both dynamicity and sparsity of these networks [21].

An alternative to the packet synchronization approaches, pulse-coupled synchronization is proposed in [22], where nodes only exchange zero-bit pulse instead of packets. The convergence properties of [22] is studied in [23] through graph theory. The dynamic and bi-directional properties of [22] is analyzed in [24]. Idle listening is introduced in [25] to reduce total energy consumption. Failing communications between vehicles are neglected, which does not cause serious impact to the result of the algorithm. Pulse-coupled synchronization [22]–[25] can be viewed as first-order phase locked loops [26], which suffer from carrier frequency offset between vehicles resulting in a constant phase lag. A pulse-coupled frequency synchronization proposed by [27] but it only synchronizes the frequency of nodes without aligning offsets. Our goals are to adapt the algorithm to vehicle network and to improve the pulse-coupled approach proposed by [24]. We apply both corrections for offset and frequency to local clocks of nodes. The simulation results show that the clock skew is reduced effectively.

[1]Cheng-Yu Han, Thomas Nowak and Alain Lambert are with Laboratoire de Recherche en Informatique (LRI), CNRS, Univ Paris-Sud, University Paris-Saclay, 91403 Orsay, France. cheng-yu.han@u-psud.fr, thomas.nowak@u-psud.fr, alain.lambert@u-psud.fr

II. Formal Model and Problem Specification

The goal of clock synchronization is to synchronize the virtual clocks of all vehicles because physical clocks of each sensor do not run at the same frequency, and these frequencies might change over time. Fig. 1 shows the virtual clocks of vehicle i and j, where the clock drift represents the difference of clock periods between vehicles. The clock skew represents the largest difference of time between vehicles. The goal is to reduce the clock skew and the clock drift as much as possible: the clocks of the vehicles should be synchronized and have the same time period.

A finite set $\mathcal{N} = \{1, 2, \ldots, n\}$ of *vehicles* communicating by message passing is considered. They are endowed with imperfect local clocks evolving in the time-base $\mathcal{T} = [0, \infty)$ of nonnegative reals.

An environment is modeled by a set of scenarios $G_t = (\mathcal{N}, \mathcal{V}_t), t \in \mathcal{T}$, where $\mathcal{N}$ is the set of vehicles and $\mathcal{V}_t$ is a set of directed edges. The relation $(i, j) \in \mathcal{V}_t$ means that vehicle i is able to receive message from vehicle j at time t. Mobility is modeled by having a (possibly different) communication graph at each time instant.

A *local algorithm* for a vehicle comprises a set of *states* $\mathcal{S}$, a set of *events* $\mathcal{E}$, and its related functions $\mathcal{F}$. All vehicles are initialized with an *initial state* $s_0 \in \mathcal{S}$. Vehicles change their state only when an event $e \in \mathcal{E}$ occurs. Vehicles modify the state considering the event e.

An *execution* of an algorithm in scenario G_t is a sequence of events $(e_k)_{k\geq 0}$, which are triggered by vehicles $i_k \in \mathcal{N}$ with its present states $s_k \in \mathcal{S}$ at a specific time $t_k \in \mathcal{T}$. The value of k is a sequence number of the event. The sequence of events are sorted by t_k. At the beginning of G_t, every vehicle triggers its first event with its initial state $s_0^{(k)}$ at time $t_k = 0$.

There are three types of event $e^{\mathrm{b}}, e^{\mathrm{m}}, e^{\mathrm{t}} \subset \mathcal{E}$. The event e^{b} represents the beginning of the local algorithm. The event e^{m} represents that a sensor receives a message. The event e^{t} represents the count down timer $\triangle t_i$ of sensor i is expired ($\triangle t_i = 0$). A vehicle i determines its next state $s_{k'}$ using the *state transition* function $\delta(s_k, e)$.

Functionalities of vehicles are involved when performing clock synchronization: a vehicle i is able to send a message to vehicle j at time t if $(i, j) \in \mathcal{V}_t$. A vehicle i produces and sends a message using its *sending function*: $\mathcal{S} \to \mathcal{M}$, which map the state s_k to a message content. A vehicle j is able to receive messages through a *receive function* when a message event e^{m} occurs. The variation of delay between sending and receiving a message is assumed to be zero. The assumption is reasonable if the message contains only few bits.

All vehicles are equipped with a *local physical clock* c_i which is a counter initiated at 0 and increasing over time subject to a clock drift bounded by parameter ϱ. Vehicles use c_i to estimate the experienced time

$$(1 - \varrho)\mu c_i \leq t \leq (1 + \varrho)\mu c_i,$$

where μ is the ideal time unit of the count down timer. All vehicles are also equipped with count down timers $\triangle c_i$. The value of $\triangle c_i$ is set using *timer set function* $\tau_i(s_k)$. Once $\triangle c_i$ is set, it starts to decrease to zero whenever c_i increases. The value of $\triangle c_i$ can be reset while counting down. When $\triangle c_i$ reaches to zero, an event $e^{\mathrm{t}} \in \mathcal{E}$ occurs. Then, $\triangle c_i$ stays at zero until it is set up again.

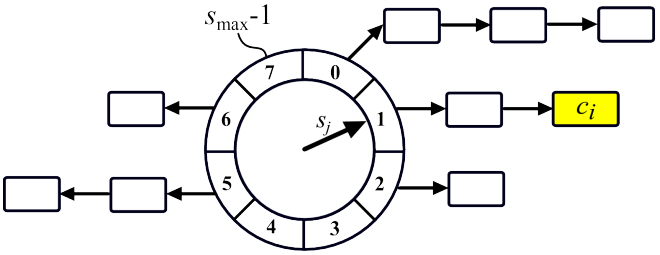


Fig. 2: When sensor i receives a pulse with a tag s_j from sensor j, it appends the local clock time c_i to the list of s_j-th slot of w_i.

Due to clock skew, one can only guarantee the time difference of $\triangle c_i$ counting to zero within an interval $[(1 - \varrho)\mu\triangle c_i, (1 + \varrho)\mu\triangle c_i]$, where μ is the time unit of the countdown timer.

A *(local) clock synchronization algorithm* is an algorithm with an additional clock set function $\gamma : \mathcal{S} \to \mathcal{T}$. An algorithm *solves clock synchronization* with precision π in scenario (G_t) if in all its executions (e_k) in G_t, we have

$$|\gamma(e(i, t)) - \gamma(e(j, t))| \leq \pi$$

for all node i and j and all times $t \in \mathcal{T}$, where $e(i, t)$ is the last state of an event of node i before time t in execution (e_k). An algorithm solve clock synchronization with precision π in an environment if it does so in each of its scenarios.

III. Time Wheel Algorithm with Drift Compensation

In this section, we introduce our proposed algorithm.

A. Synchronization functions

We start by describing the general round-structure that our algorithm operates in.

Vehicles start in round $k = 0$. After vehicle i broadcasts a pulse in round $k - 1$, the vehicle waits for a period of time $T_i(k)$, which is subject to its local clock drift. Because the durations $T_i(k)$ and the round start times may not be exactly the same at different vehicles, pulses are not perfectly synchronized with each other. Therefore, corrections must be computed using information of previous pulses from neighboring vehicles.

In [24], round numbers are assumed to be known for all vehicles, which is not always realistic, because rounds estimated by vehicles can be faster or slower. In our new algorithm, vehicle i broadcast a round value $s_i \in \mathbb{Z}, 0 \leq s \leq s_{\max} - 1$. The value s_i serves a similar role as k, which increase every round, except when s_i reaches $s_{\max}$, it return to zero for the next round.

Algorithm 1 Vehicle operations

1: Procedure for vehicle i
2: **At initialization**
3: $w_i = \text{TimeWheel}(s_{\max})$
4: $s_i = 0$
5: $c_i = 0$
6: sending_function(s_i)
7: **When receiving message** m
8: Obtain s_j from m
9: Append c_i to the list in s_j-th slot
10: $\tau(k+1) =$ timer_set_function(w_i, $\tau(k)$)
11: $\triangle c = \tau(k+1) - c_i$
12: **When timer expires**
13: $w[(s_i + \lfloor s_{max}/2 \rfloor) \bmod s_{max}]$.clear()
14: $s_i = (s_i + 1) \bmod (s_{\max})$
15: $k_i = k_i + 1$
16: sending_function(s_i)

Vehicle i uses a *time wheel* w_i to record s_j from neighboring vehicle j. Fig. 2 shows the structure of the time wheel w_i. A time wheel of sensor i has $s_{\max}$ slots. The pointer in the center of the time wheel represents the value of s_j broadcasted by sensor j. When sensor i receives a pulse with a tag s_j from sensor j, it appends the local clock time c_i to the list of s_j-th slot of w_i. The data in w_i will be used to estimate the times for the next round.

Each vehicle runs the pseudo-code given in Algorithm 1. The code is divided into three parts: initial function, receive function, and state transition function. These three parts are not executed in sequence. Instead, they are triggered by different events. The initial function is executed once when the vehicles start to synchronize the clock. The receive function is executed whenever s_j is received. The state transition function is executed when a new round starts.

a) Initial function: All vehicles initialize an empty time wheel w_i with $s_{\max}$ slots (line 3). Vehicles also reset s_i (line 4) and c_i (line 5) to zero. The variable c_i represents ticks of the physical clock started at initialization, which increases subject to the clock drift. A vehicle also broadcasts s_i to its neighbors (line 6).

b) Receive function: When vehicle i receives a message from sensor j, with a tag s_j (line 8). Vehicle i appends the current local time c_i to the list of the s_j-th slot (line 9). Recall that the delay variation between broadcasting and receiving a pulse is assumed to be negligible, i.e., $c_i \approx t_j(k_j)$. Vehicles also schedule their next state transition by setting their local timer to $\tau_i(k)$ (line 10). Details can be found in section III-B. Countdown timer $\triangle t$ is set to trigger the state transition function when the timer reaches zero (line 11).

c) State transition function: When the timer reaches zero, the vehicle executes the state transition function. The vehicle transitions to the next round: First, the vehicle clears the slot farthest from the current slot s_i (line 13). Second, it updates s_i and k_i (line 14, 15). At the end, it broadcasts s_i to its neighboring vehicles (line 16).

In previous work, the rounds are assumed known by

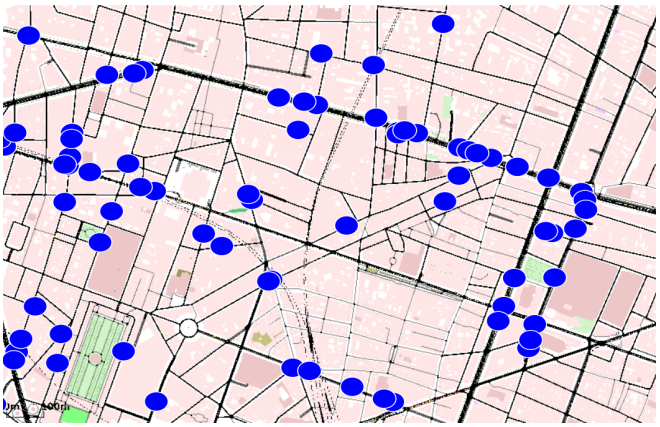


Fig. 3: Static vehicle network simulation environment

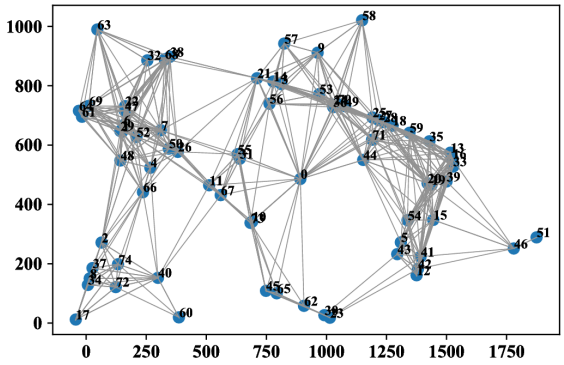


Fig. 4: Static vehicle network

vehicles and the pulses in the same round is assumed to have only small difference, In this work, the time wheel is applied to improve the robustness of the algorithm, so the assumption is no longer needed.

B. Timer set function

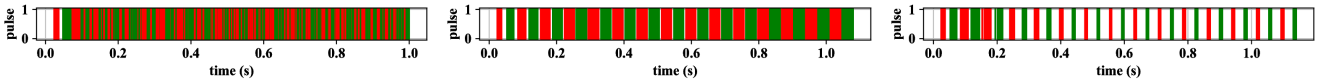
The timer set function in Algorithm 1 estimates $\tau(k)$ according to the following formula:

$$\tau_i(k+1) = \tau_i(k) + R + \text{corr}_i^1 + \text{corr}_i^2, \tag{1}$$

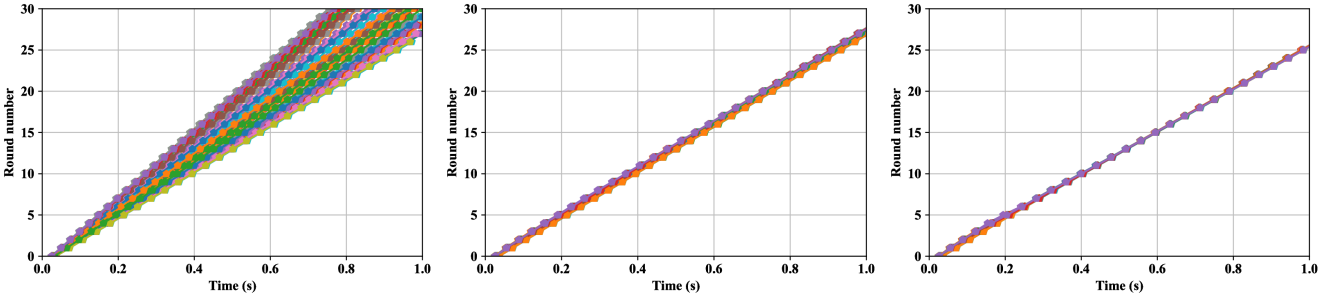
where R is a constant represents a time period between next and present round. All vehicle uses same value of R, but the time period $T_i(k+1)$ may be differ due to frequency variation of local clocks:

$$(1-\varrho)\mu R \leq T_i(k+1) \leq (1+\varrho)\mu R \tag{2}$$

where corr_i^1 is the correction for the offset of each pulse, and corr_i^2 is the correction for clock drift. Each vehicle only communicates through pulses with the correction time and addition round information c_i. The overall benefit of the method is that vehicles do not need to identify their neighboring vehicle and their pulses, so the message is short and robust.



(a) waveform of the pulses



(b) timing of pulses at each round

Fig. 5: Simulation result shows the comparison between broadcasted pulses using (1) no algorithm (left), (2) corr_i^1 (middle) and using (3) $\text{corr}_i^1 + \text{corr}_i^2$ (right).

1) Offset compensation: The term corr_i^1 is used for correcting the offset for all the vehicles in previous work [24]. corr_i^1 is a time difference between the average pulse of the neighboring vehicles and the pulse of vehicle i itself in round k:

$$corr_i^1 = \frac{1}{|\text{In}_i(k+1)|} \sum_{j \in \text{In}_i(k+1)} (\tau_j(k) - \tau_i(k)) \quad (3)$$

where $\sum_{j \in \text{In}_i(k+1)} (\tau_j(k) - \tau_i(k))$ in (3) is calculated by averaging the values in the $k\%(s_{\max}+1)$-th slot of wheel w_i.

With offset compensation, the pulses already fulfills $\forall k \geq 0, \max_{i,j} |t_i(k) - t_j(k)| \leq \sigma^k$, where the σ_k is the maximum clock skew at each round. The variables $t_i(k)$ and $t_j(k)$ represent the beginning time of sensor i and j in the k-th round.

2) Drift compensation: After the offset compensation algorithm is applied, all the previous pulse virtually have the same offset, i.e., $t_i(k_i) + \text{corr}_i^1 \approx t_j(k_j) + \text{corr}_j^1$. At this point, it is only necessary to compensate for possible drift errors, i.e., to find a common clock cycle T^g and let $T^g \approx T_i(k_i+1) + \text{corr}_i^2 \approx T_j(k_j+1) + \text{corr}_j^2 \approx \ldots$. Instead of exchanging the timing information and estimating the global clock cycle $T^i(k_j)$ between vehicles, we only use local variables of vehicles to perform the correction.

The goal of drift compensation is to let vehicles agree on a global clock cycle T^g. The main idea is to use the $\tau_i(k)$ of the vehicle itself. We obtain an approximation of the common clock cycle using

$$T^g \approx \mu \frac{\tau_i(k_i)}{k}, \quad (4)$$

where k_i and l_i satisfy $k_i, l_i \in \mathbb{R}, 0 \leq l_i \leq k_i$, so that T^g is an average clock cycle of the vehicle i. In order to correct $T_i(k_i+1)$, the correction of drift compensation is computed:

$$corr_i^2 = \frac{\tau_i(k_i)}{k} - R \quad (5)$$

The correction corr_i^2 waits l_i rounds until the clock skew is small enough for all the vehicles. If l_i is set to 0 and the connectivity of the communication graph is low, there is a possibility that corr_i^2 increases the divergence of the pulses. Our experiment discovers that $l_i \geq 5$ is sufficient for all the experimental cases.

IV. Simulations

We simulate a vehicle network thanks to the Simulation of Urban MObility (SUMO). The simulation environment is the 2nd district of Paris, downloaded from OpenStreetMap [28].

A. Static Vehicle Network

Fig. 3 shows the simulation environment, where $n = 75$ vehicles are randomly placed with an average number of 2.84 neighbors per vehicle. Each vehicle has different sensing range to detect the pulses from neighboring vehicles. Vehicles can only sense the neighbors within the distance $r_i \in \mathbb{R}$, which is bounded within a range: $r(1-\epsilon) \leq r_i \leq r(1+\epsilon)$. The average of sensing range for all the vehicles is $r = 300$ (m) and $\epsilon = 0.5$. And the number of slots of timewheel is $s_{\max} = 2$.

Fig. 4 shows the vehicle nodes and their connectivity. The size of the scene is 1000×2000 m^2. We set $T_i(k) = R(1 + \sigma_i + \sigma_k)$ with $|\sigma_i| \leq 20\%$ and $|\sigma_k| \leq 1\%$ and $R = 0.03$ second. σ_i is only applied once when a vehicle is initialized. σ_k changes in each round, which affects the frequency of clock every round. A realistic value of σ_k is $10^{-6}\%$. We setup all the vehicles waking up at time 0 and starting to broadcast at time $T_i(0)$.

We compare the three different algorithms in the exactly same environment:

1) without any clock synchronization
2) clock synchronization using only corr_i^1
3) clock synchronization using $\text{corr}_i^1 + \text{corr}_i^2$

The second algorithm is our improvement of the work published by [24] and the third is our new proposed technique.

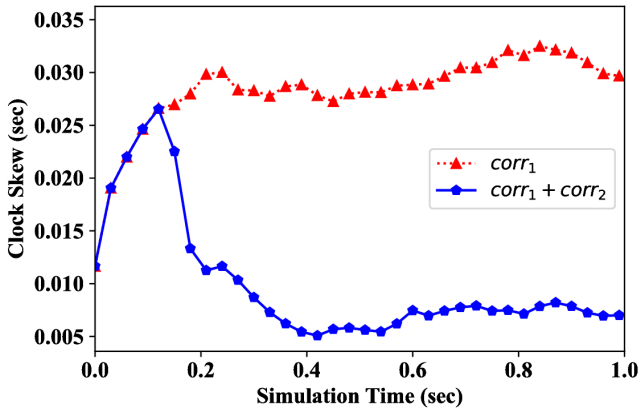


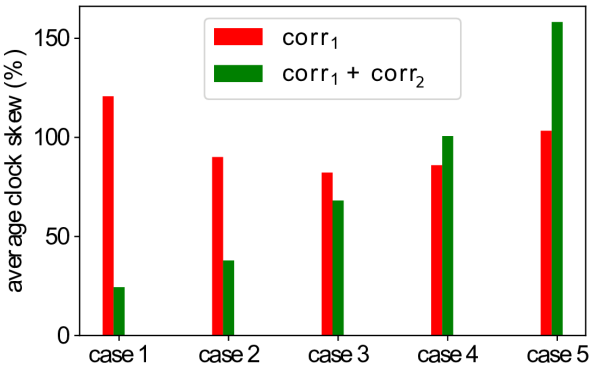
Fig. 6: Clock skew comparison between corr_1 (red) and $\text{corr}_1 + \text{corr}_2$ (blue)

We take corr_i^2 into account after at the sixth round. Fig. 5 shows the comparison of these three algorithms. Fig. 5 (a) shows the waveform of the pulses for all vehicles from 0 to 1 second. The x-axis indicates time in seconds and the y-axis shows a value 1 whenever any pulse is broadcasted, otherwise it shows 0. The green and red colors indicate the different rounds: $s_i = 0$ and $s_i = 1$. The color lines are overlapped in the middle figure of Fig. 5 (a), so the actual clock skew is larger than the appearance. Fig. 5 (b) shows the timing of pulses for each round. Dots linked by lines with different colors indicate that the pulses are broadcasted by the same vehicle. One can observe that if no clock synchronization technique is applied, the pulses for each round diverge (column 1 in Fig. 5). Pulses in the same round for algorithm "corr_i^1" are successfully trapped within an interval, but the interval will not reduce as time goes on (column 2 in Fig. 5). In contrast, pulses from the method $\text{corr}_i^1 + \text{corr}_i^2$ are broadcasted more orderly than corr_i^1 as time goes on (column 3 in Fig. 5).

Fig. 6 shows the clock skew for each round. We calculate clock skew by considering the nearest pulse from all vehicles in the same round k to the target vehicle. Clock skew for first five rounds are the same, because $\text{corr}_1 + \text{corr}_2$ includes the term $\text{corr}_2 = 0$. As soon as corr_2 is enabled, the clock skew reduces significantly. The clock skew of $\text{corr}_1 + \text{corr}_2$ is approximately 7 times smaller than the clock skew of corr_1 after both are stable.

B. Clock Drift Analysis

In this experiment, we analyze the methods in situations where σ_i and σ_k are different to verify the robustness of corr_i^1 and $\text{corr}_i^1 + \text{corr}_i^2$. In order to test the edge cases, the total variation of $\sigma_i + \sigma_k$ is $\pm 20\%$ of the constant R. Fig. 7 shows the five different situations. The scene is exactly the same except σ_i and σ_k is changed. In this fifth simulation, the error for method corr_i^1 becomes smaller when σ_i decrease, which shows that it can handle random uncertainty of clock drift σ_k better than $\text{corr}_i^1 + \text{corr}_i^2$. In contrast, the error for method $\text{corr}_i^1 + \text{corr}_i^2$ increase when σ_k increases. It shows



	case1	case2	case3	case4	case5
$\sigma_{i,\max}$	$\leq 20\%$	$\leq 15\%$	$\leq 10\%$	$\leq 5\%$	$\leq 0\%$
$\sigma_{k,\max}$	$\leq 0\%$	$\leq 5\%$	$\leq 10\%$	$\leq 15\%$	$\leq 20\%$

Fig. 7: Experiment 2

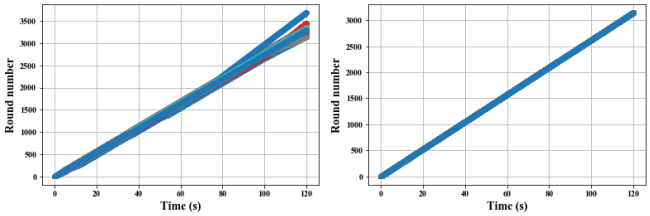


Fig. 8: Time of each rounds of a dynamic vehicle network using different algorithms: (left) corr_1 (right) $\text{corr}_1 + \text{corr}_2$

that large amount of additional random clock drift can make $\text{corr}_i^1 + \text{corr}_i^2$ synchronize worse than corr_i^1 alone. Notice that we test the edge cases here, in normal cases, σ_k is smaller than $10^{-4}\%$

C. Moving Vehicle Network

Vehicles are initially setup as Fig. 3, but start to move with the maximum speed on the road. The acceleration and the deceleration of each vehicle is 2.6 (m/s^2) and 4.5 (m/s^2), respectively. Vehicles slow down when they turn and stop when traffic lights are red. There are 214 traffic lights on the map. The start and end points of vehicles are randomly generated by SUMO.

Fig. 8 shows the comparison of the two methods (corr_1 and $\text{corr}_1 + \text{corr}_2$). Fig. 8 (left) shows the rounds and the pulses using only corr_1. Each color represents a vehicle and each dots represents both the time and the round number of pulses. Fig. 8 (right) shows the round of the pulses using both corr_1 and corr_2. The color of points overlaps so only single color can be seen. The pulses are aligned better with algorithm $\text{corr}_1 + \text{corr}_2$ than corr_1. Pulses using $\text{corr}_1 + \text{corr}_2$ converge faster than corr_1.

D. Pulse Frequency and Convergence Rate

In this section, different frequencies of pulse are considered. With the same dynamic vehicle network, if the vehicles exchange pulses more frequently, then the clock skew tends

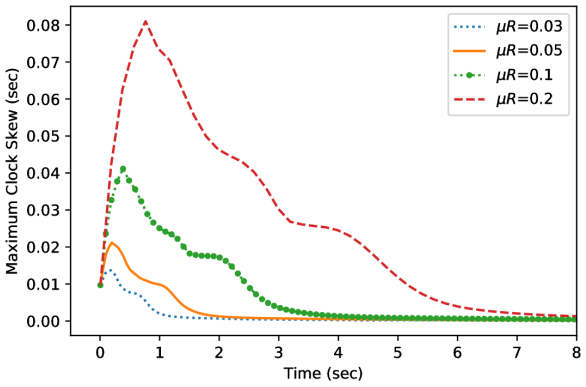


Fig. 9: Simulation result for different frequency of pulse

to converge faster. Fig 9 illustrates four simulations where the scene setup is exactly the same as Fig 3 (except the default time period between two pulses μR which is different). The variable R is the number of ticks of the internal physical clock between pulses, and μ is the ideal time period for a single tick of the internal physical clock of a vehicle. If the value of the μR is small, vehicles exchange pulses rapidly and pulses tend to converge faster than the simulation that μR is large. However, pulses exchange frequently cost extra energy consumption considering the same time period. The parameter μR can be chosen to balance the convergence rate and the energy consumption. In the vehicle network, all pulses in the same round converge regardless of μR. There is also a possibility to slow down the pulse exchange after the clock synchronization is converged.

V. Conclusion

This work improves the pulse-coupled synchronization by introducing a time-wheel algorithm and drift compensation. The time-wheel algorithm enables vehicles to exchange the information of the rounds during clock synchronization. Drift compensation applies an additional correction to clock skew. The simulation results show that the proposed algorithm performs better than the compared algorithm, especially when considering a highly dynamic wireless vehicle network.

For future work, we intend to evaluate the algorithm using practical wireless vehicle network. The first step will be to select an appropriate communication hardware and to closely simulate it in a large scale network.

References

[1] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Conference on Embedded networked sensor systems*. ACM, 2003, pp. 138–149.

[2] S. A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma, "Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing," in *Conference on Advanced Information Networking and Applications*, vol. 2. IEEE, 2007, pp. 113–120.

[3] A. A. Syed, J. S. Heidemann, *et al.*, "Time synchronization for high latency acoustic networks." in *Infocom*, vol. 6, 2006, pp. 1–12.

[4] K. Sun, P. Ning, and C. Wang, "Fault-tolerant cluster-wise clock synchronization for wireless sensor networks," *Transactions on Dependable and Secure Computing*, vol. 2, no. 3, pp. 177–189, 2005.

[5] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad hoc networks*, vol. 3, no. 3, pp. 281–323, 2005.

[6] R. Solis, V. S. Borkar, and P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *Conference on Decision and Control*. IEEE, 2006, pp. 2734–2739.

[7] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384–397, 2005.

[8] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 149–154, 2003.

[9] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin, "Coping with irregular spatio-temporal sampling in sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 125–130, 2004.

[10] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.

[11] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE wireless communications*, vol. 11, no. 6, pp. 6–28, 2004.

[12] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *IEEE Symposium on Communications and Vehicular Technology*, 2011, pp. 1–6.

[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[14] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of IEEE Infocom*, vol. 3, 2002, pp. 1567–1576.

[15] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[16] F. Cristian, "Probabilistic clock synchronization," *Distributed computing*, vol. 3, no. 3, pp. 146–158, 1989.

[17] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 933–940, 1987.

[18] L. Schenato and F. Fiorentin, "Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, 2011.

[19] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in wsns: A maximum-value-based consensus approach," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 660–675, 2014.

[20] R. Carli and S. Zampieri, "Network clock synchronization based on the second-order linear consensus algorithm," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 409–422, 2014.

[21] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, 2011.

[22] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on computers*, vol. 55, no. 2, pp. 214–226, 2006.

[23] O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, p. 057054, 2007.

[24] M. Függer, T. Nowak, and B. Charron-Bost, "Diffusive clock synchronization in highly dynamic networks," in *Conference on Information Sciences and Systems*. IEEE, 2015, pp. 1–6.

[25] Y. Wang, F. Nunez, and F. J. Doyle, "Energy-efficient pulse-coupled synchronization strategy design for wireless sensor networks through reduced idle listening," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5293–5306, 2012.

[26] O. Simeone, U. Spagnolini, G. Scutari, and Y. Bar-Ness, "Physical-layer distributed synchronization in wireless networks and applications," *Physical Communication*, vol. 1, no. 1, pp. 67–83, 2008.

[27] N. Varanese, U. Spagnolini, and Y. Bar-Ness, "Distributed frequency-locked loops for wireless networks," *IEEE Transactions on Communications*, vol. 59, no. 12, pp. 3440–3451, 2011.

[28] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org , 2017.