CHALMERS

Chalmers Publication Library

Random Broadcast Based Distributed Consensus Clock Synchronization for Mobile Networks

This document has been downloaded from Chalmers Publication Library (CPL). It is the author´s version of a work that was accepted for publication in:

IEEE Transactions on Wireless Communications (ISSN: 1536-1276)

Citation for the published paper:
Wanlu, S. ; Ström, E. ; Brännström, F. et al. (2015) "Random Broadcast Based Distributed Consensus Clock Synchronization for Mobile Networks". IEEE Transactions on Wireless Communications

Downloaded from: http://publications.lib.chalmers.se/publication/213489

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source. Please note that access to the published version might require a subscription.

Chalmers Publication Library (CPL) offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all types of publications: articles, dissertations, licentiate theses, masters theses, conference papers, reports etc. Since 2006 it is the official tool for Chalmers official publication statistics. To ensure that Chalmers research results are disseminated as widely as possible, an Open Access Policy has been adopted.
The CPL service is administrated and maintained by Chalmers Library.

(article starts on next page)

Random Broadcast Based Distributed Consensus Clock Synchronization for Mobile Networks

Wanlu Sun, Erik G. Ström, *Senior Member, IEEE,* Fredrik Brännström, *Member, IEEE,* and Mohammad Reza Gholami, *Member, IEEE*

***Abstract*—Clock synchronization is a crucial issue for mobile ad hoc networks due to the dynamic and distributed nature of these networks. In this paper, employing affine models for local clocks, a random broadcast based distributed consensus clock synchronization algorithm is proposed. In the absence of transmission delays, we theoretically prove the convergence of the proposed scheme, which is further illustrated by numerical results. In addition, it is concluded from simulations that the proposed scheme is scalable and robust to transmission delays as well as different accuracy requirements.**

***Index Terms*—Mobile ad hoc networks, distributed synchronization, consensus algorithm, random broadcast, convergence.**

I. Introduction

RECENTLY, ad hoc networks have emerged as an interesting and important research area. As this type of network consists of many small computing devices and its communication services are based on self-organizing, clock synchronization becomes critical for many applications. For instance, coordination of actions across a distributed set of nodes requires an accuracy of millisecond level; an order of microsecond accuracy is needed for time-division multiple access (TDMA) technique; and target localization requires nanosecond accuracy. Meanwhile, there is also an increasing demand for mobile networks (e.g., vehicular networks), where the mobility complicates the clock synchronization due to the time evolving topologies.

A. Related Work

There has been extensive research on clock synchronization in the context of ad hoc networks during the last few years. Depending on whether the reference nodes are needed or not, existing protocols can be mainly classified into two categories: reference-based clock synchronization and distributed clock synchronization. In reference-based clock synchronization [1], [2], one node is elected as the reference node and a spanning tree is built through the network. All the other nodes are required to synchronize to the reference node by adjusting their own clocks based on the timing messages received from their parents. This mechanism is sensitive to changing topologies and node failures, and therefore not suitable for mobile networks. On the other hand, in distributed clock synchronization, all nodes implement the same algorithm individually without relying on a network hierarchy [3]–[13]. The distributed nature can often result in improved robustness to node failures and mobility in dynamic networks.

In distributed clock synchronization, to utilize the broadcast nature of wireless medium, nodes broadcast timing messages which contain the timestamps recorded by the clock of the transmitter. These messages are in turn used to adjust the clocks of the receivers. We assume that the broadcasted messages can only reach the neighbors of the transmitter, in which case the network is not necessarily fully connected.

Distributed synchronization algorithms require timing messages from neighboring nodes, which can be used in two different ways.

- Simultaneous update [7], [10], [11]: each node first collects timing messages from its immediate neighbors, and then adjusts its local clock by using these messages simultaneously.
- Sequential update [3]–[6], [8], [9], [12], [14]: each node sequentially update its clock whenever it receives a timing message.

In this paper, we consider sequential updates with the assumption of a random access mechanism. With a random access mechanism, a node can broadcast at any time in any order. A widely used random broadcast scheme is contention based transmission, where nodes contend for transmission opportunities at the beginning of each synchronization round (SR). SRs are repeated with some predetermined periodicity, and nodes are thereby granted some fairness in accessing the wireless medium. Due to its applicability in distributed networks, this mechanism is the technique specified for clock synchronization in the IEEE 802.11 standard [3] and has been used in some other works [4]–[8] as well. As described in [3], the transmission protocol of timing messages assumes that nodes are loosely synchronized to the beginning of each SR. In this paper, we will also use a contention based transmission protocol.

Wanlu Sun, Erik G. Ström, and Fredrik Brännström are with the Division of Communication Systems, Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden. E-mail: {wanlu, erik.strom, fredrik.brannstrom}@chalmers.se

Mohammad Reza Gholami is with the ACCESS Linnaeus Centre, Signal Processing Department, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. E-mail: mohrg@kth.se

This work has been supported in part by SAFER-Vehicle and Traffic Safety Centre, Project A19 and by the Swedish Research Council project 2011-5824. Part of this work has been performed in the framework of the FP7 project ICT-317669 METIS, which is partly funded by the EU. The authors would like to acknowledge the contributions of their colleagues in METIS, although the views expressed are those of the authors and do not necessarily represent the project. The calculations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at C3SE.

Part of this work was presented at the 2013 IEEE GLOBECOM workshop.

When assuming random broadcast mechanism for message transmission, the authors in [3]–[6] propose different synchronization schemes based on converge-to-max principle, where a node eventually synchronizes to the nodes with faster clocks. A simple converge-to-max protocol, called timing synchronization function (TSF), is presented in [3]. Based on the TSF, various modifications have been made to handle its limitation of scalability and infeasibility in multihop networks [4]–[6]. For example, the modified automatic selftime-correcting procedure (MASP) scheme is proposed in [6], which outperforms the other methods. Nevertheless, as addressed in [15], a common problem for all converge-to-max schemes is the contradiction between the *fastest node asynchronism* and the *time partitioning*[1].

With a random broadcast mechanism, the authors in [8] propose a distributed consensus based protocol for clock synchronization, which is referred as ATS. In the ATS scheme, an internal common time scale, which does not need to be the maximum, is achieved in the network through communications among neighboring nodes. In practice, however, clock frequencies may be over-adjusted due to the unawareness of clock updates at the transmitter or receiver, and thus consensus will not be achieved. Additionally, the authors in [14] propose a consensus based clock synchronization (CoSyn) algorithm for the random broadcast protocol, whereas the convergence is not rigorously proved.

B. Contributions

In this paper, based on a practical random broadcast mechanism, a novel distributed consensus clock synchronization algorithm is proposed for dynamic networks. It is fully distributed in the sense that all the nodes independently execute the same algorithm without the need of a network hierarchy, and is thus robust to node failures and changing topologies. The key feature of the proposed scheme is that it distinguishes between two different updates—partial updates and complete updates—for different situations. In this way, the proposed scheme can both avoid the problem of frequency over-adjustment and improve the speed of synchronization error decrease. In the absence of transmission delays, we theoretically prove the convergence of the proposed method, which is further demonstrated by numerical results. Moreover, by utilizing a threshold for the clock update, the proposed scheme reveals robustness even when transmission delays are present.

II. System Model and Problem Formulation

A. Notation

Lowercase and uppercase letters, e.g., x and X, represent scalars, lowercase boldface letters, e.g., $\boldsymbol{x}$, designate column vectors where $[\boldsymbol{x}]_i$ indicates the ith element, and uppercase boldface letters, e.g., $\boldsymbol{X}$, denote matrices where $[\boldsymbol{X}]_{ij}$ denotes the (i,j)th element. Also, $\mathbf{1}$ and $\mathbf{0}$ represent the all-ones column vector and zero column vector, respectively. The superscript $(\cdot)^{\mathsf{T}}$ stands for transposition. Sets are denoted by calligraphic letters $\mathcal{X}$ and their cardinalities are denoted by $|\mathcal{X}|$.

B. Network Model

We consider a network represented by a directed graph $\mathcal{G}(\ell) = (\mathcal{V}, \mathcal{E}(\ell))$, where the vertex set $\mathcal{V} = \{1, 2, ..., N\}$ contains N mobile nodes and the edge set $\mathcal{E}(\ell)$ is defined as the set of available directed communication links at the discrete time index ℓ, i.e., $(i,j) \in \mathcal{E}(\ell)$ if node j sends information to node i during the ℓth SR. $\forall i \in \mathcal{V}$, we use the convention that $(i,i) \in \mathcal{E}(\ell)$ for all ℓ. The notation $\mathcal{V}_i(\ell) = \{j|(i,j) \in \mathcal{E}(\ell)\}$ denotes the set of neighbors of node i during the ℓth SR.

C. Clock Model

Each node in the network is equipped with a physical clock that has its frequency and offset. Here, we assume an affine function for the physical clock model. In this way, the physical clock of node i is

$$T_i(t) = f_i t + \theta_i, \quad \forall i \in \mathcal{V}, \tag{1}$$

where t is the perfect time, f_i indicates the physical clock frequency, and θ_i denotes the physical clock offset. Note that f_i and θ_i are both determined by the physical clock and cannot be measured or adjusted.

Remark 1. Here we assume f_i is constant over time. In practice, however, f_i can vary over time due to the drifts introduced by aging effects and environmental factors such as temperature [16]. Nevertheless, as pointed out in [5], [17], [18], the drift resulted from aging usually acts much slower than the update rate of synchronization schemes, and thus can be safely ignored. On the other hand, we will evaluate the impact of the drift induced by temperature variation by numerical simulations. If the environmental factors do cause significant instability to physical clock frequencies, the synchronization procedures should be implemented more frequently to keep track of the clock parameters.

Besides, each node i also maintains a logical clock, whose value is denoted by $C_i(t)$ and can be modified. The logical clock $C_i(t)$ represents the synchronized time of node i, which is a function of the current physical clock value $T_i(t)$. In this paper, we use an affine model for $C_i(t)$:

$$C_i(t) = \alpha_i T_i(t) + \beta_i \tag{2}$$

$$= \alpha_i f_i t + \alpha_i \theta_i + \beta_i \tag{3}$$

$$= \hat{f}_i t + \hat{\theta}_i, \tag{4}$$

where α_i ($\alpha_i > 0$) and β_i are control parameters updated by the synchronization algorithm, and

$$\hat{f}_i \triangleq \alpha_i f_i \tag{5}$$

[1]Since a node only synchronizes to a faster node, the clock value of the fastest node (a node with the greatest clock value in the network) will keep drifting away from other nodes, unless it becomes the transmitter. This problem is called the fastest node asynchronism problem, which can be reduced by giving higher priority to the transmissions of the node with a faster clock. Nevertheless, different priorities might result in the time partitioning problem, where the clock values in two groups of nodes can keep on drifting away from each other, even though they are connected.

$$\hat{\theta}_i \triangleq \alpha_i \theta_i + \beta_i \tag{6}$$

represent the logical clock frequency and logical clock offset. The initial values of α_i and β_i are set to $\alpha_i = 1$ and $\beta_i = 0$, respectively. In this way, the goal is to synchronize the clocks in the entire network such that the logical clocks of different nodes have the same (or very close) values for any instant of perfect time t.

D. Problem Formulation

In this study, rather than synchronizing all clocks to a real reference clock, we aim at attaining an internal consensus between logical clocks through local interactions between nodes. Namely, we say that the consensus is achieved if

$$\lim_{t\to+\infty} \frac{C_i(t)}{C_{\rm v}(t)} = 1, \quad \forall i \in \mathcal{V}, \tag{7}$$

where

$$C_{\rm v}(t) = f_{\rm v} t + \theta_{\rm v}, \quad f_{\rm v} > 0, \tag{8}$$

denotes the virtual consensus clock. For each node i, the asymptotic consensus (7) is equivalent to concurrently achieving the following two consensus equations:

$$\lim_{t\to+\infty} \hat{f}_i = f_{\rm v}, \tag{9}$$

$$\lim_{t\to+\infty} \hat{\theta}_i = \theta_{\rm v}. \tag{10}$$

Note that $f_{\rm v}$ and $\theta_{\rm v}$ do not need to be the average value of $\{f_1, f_2, ..., f_N\}$ and $\{\theta_1, \theta_2, ..., \theta_N\}$, respectively. Their values are decided by $\mathcal{E}(\ell)$, $\{f_1, f_2, ..., f_N\}$ and $\{\theta_1, \theta_2, ..., \theta_N\}$ together. In fact, the values of $f_{\rm v}$ and $\theta_{\rm v}$ are not important, since what really matters is that all clocks converge to one common function of time.

E. Transmission of Synchronization Messages

In this paper, we consider a widely used random broadcast scheme, i.e., the contention based broadcast mechanism. More specifically, each node [3]:

1) at the beginning of each SR, calculates a random delay that is uniformly distributed in the range between zero and 2 × aCWmin × aSlotTime (which are constants and specified in [3]);

2) waits for the period of the random delay while decrementing the random delay timer;

3) cancels the remaining random delay and the pending transmission if a timing message arrives before the random delay timer has expired or;

4) sends a timing message when the random delay expires.

Remark 2. Due to the hidden node problem, it is possible for one node to receive multiple messages during one SR. In this case, the node will just keep the first received packet and discard the later packets. In other words, $\mid \mathcal{V}_i(\ell) \mid$ can only be 1 or 2 for all $i \in \mathcal{V}$.

Remark 3. The use of the random broadcast mechanism, where the information flow is in one direction, explains why we consider a directed graph $\mathcal{G}(\ell)$ in Section II-B. Note that $\mathcal{G}(\ell)$ might not be connected at each SR.

Remark 4. As in most of the literature (e.g., [1], [2], [7]–[13]), MAC layer time stamping is utilized to largely reduce the effects of transmission delays. Here, MAC layer time stamping means that the current timestamp is written into the message payload right before the first bit of the packet is sent to the physical layer at the transmitter, and the timestamp at the receiver side is recorded right after the first bit has arrived at the MAC layer.

III. The Proposed Synchronization Algorithm: RBDS

In this section, we propose a novel Random Broadcast based Distributed consensus clock Synchronization (RBDS) scheme, where the logical clock is adjusted by partial updates or complete updates to achieve consensus. We take convergence as well as error decrease speed into account for the algorithm design. Section III-A derives the basic design principles of the RBDS scheme. Then the complete procedures of the RBDS scheme are summarized in Section III-B. Finally, the convergence analysis is elaborated by Theorem 3 in Section III-C.

A. Design Principles of the RBDS Scheme

We first introduce some graph theory terminology [19], which will be used in this paper.

- A vertex i of a directed graph is a **root** of a graph if for each other vertex j of this graph, there is a path from i to j.
- A **rooted graph** is a graph which possesses at least one root.
- By the **composition** of two directed graphs $\mathcal{G}(p)$, $\mathcal{G}(q)$ with the same vertex set $\mathcal{V}$, we mean the graph $\mathcal{G}(p)\circ\mathcal{G}(q)$ with the same vertex set $\mathcal{V}$ and edge set defined such that (i, j) is an edge of $\mathcal{G}(p) \circ \mathcal{G}(q)$ if and only if for some vertex r, (r, j) is an edge of $\mathcal{G}(q)$ and (i, r) is an edge of $\mathcal{G}(p)$.
- A finite sequence of directed graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$,..., $\mathcal{G}(q)$ with the same vertex set is **jointly rooted** if the composition $\mathcal{G}(q) \circ \mathcal{G}(q-1) \circ \cdots \circ \mathcal{G}(1)$ is rooted.
- An infinite sequence of graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$, ... with the same vertex set is **repeatedly jointly rooted by sequences of length** q if there is a positive finite integer q for which each finite sequence $\mathcal{G}(qm+1)$, $\mathcal{G}(qm+2)$, ..., $\mathcal{G}(qm+q)$ for all $m = 0, 1, ...$, is jointly rooted.

The crucial result upon which the algorithm design depends is [19, Theorem. 1], which is restated here.

Theorem 1. *Let $\boldsymbol{x}^{(0)}$ be fixed. For any trajectory of the system determined by the following equation (11)*

$$x_i^{(\ell+1)} = \frac{1}{\mid \mathcal{V}_i(\ell) \mid} \left(\sum_{j\in\mathcal{V}_i(\ell)} x_j^{(\ell)} \right), \tag{11}$$

where $x_i^{(\ell)}$ represents information state associated with node i and ℓ is a discrete-time index, along which the sequence of

graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$, ..., $\mathcal{G}(\ell)$, ... is repeatedly jointly rooted by sequences of length q, there is a constant $x_{\rm ss}$ for which

$$\lim_{\ell\to+\infty} \boldsymbol{x}^{(\ell)} = x_{\rm ss}\mathbf{1}, \tag{12}$$

where the limit is approached exponentially fast, and $\boldsymbol{x}^{(\ell)} = \left[x_1^{(\ell)}, ..., x_N^{(\ell)}\right]^{\mathsf{T}}$ is a vector collecting the information state of all nodes.

In our settings, suppose node $i \in \mathcal{V}$ receives a timing message from one neighbor node $j \in \mathcal{V}$. Based on Theorem 1, in the absence of delays, if the 'repeatedly jointly rooted' condition is satisfied and if node i updates its logical frequency and logical offset as

$$\hat{f}_i^{(\ell+1)} = \frac{1}{2}\left(\hat{f}_i^{(\ell)} + \hat{f}_j^{(\ell)}\right), \tag{13}$$

$$\hat{\theta}_i^{(\ell+1)} = \frac{1}{2}\left(\hat{\theta}_i^{(\ell)} + \hat{\theta}_j^{(\ell)}\right), \tag{14}$$

respectively, then the consensus (9) and (10) will be achieved with exponential speed. Note that, as explained in Remark 2, we only have the access to at most one neighbor node in the update procedure.

In practice, however, the implementation of (13) and (14) is not straightforward. There are mainly two difficulties.

Firstly, due to the lack of information about f_i and θ_i, we can neither obtain $\hat{f}_i$ or $\hat{\theta}_i$ nor tune them directly. What we can do is to modify the control parameters α_i and β_i by utilizing the logical clock values, which will then adjust $\hat{f}_i$ and $\hat{\theta}_i$ automatically through (5) and (6).

Secondly, in order to execute the update of the logical frequency as in (13), the following two requirements are sufficient: *1)* node i receives at least two timing messages from node j; *2)* logical clocks of both node i and node j are not adjusted between the two receptions. We will show this sufficiency later. Obviously, the conditions can be satisfied if we make node i not to adjust its logical clock (i.e., not update its control parameters) until it receives the second message from node j. However, this method will result in fairly slow convergence speed, especially for dense networks. Therefore, in our algorithm, we propose a novel update rule, by distinguishing two different updates, to guarantee that (13) is satisfied even when the node i's logical clock have been updated between two consecutive messages from node j. Nonetheless, this new update rule will give rise to the result that the update rule (14) cannot be perfectly satisfied in the proposed scheme, which will be proved later by Theorem 2.

In the following, we will explain the proposed algorithm from the perspective of node i. For easier reading, the explanations of some used symbols are summarized in Table I.

If node i does not receive any timing message during the ℓth SR, there is no update in its logical clock, i.e.,

$$\begin{cases} \alpha_i^{(\ell+1)} = \alpha_i^{(\ell)} & (15) \\ \beta_i^{(\ell+1)} = \beta_i^{(\ell)}, & (16) \end{cases}$$

and we define $\Delta_i^{(\ell)} \triangleq 0$ for this case. Otherwise, i.e., the node i receives timing messages from its neighbors during the ℓth SR, it will update either β_i or both β_i and α_i according to one of two update rules: the partial update rule or the complete update rule.

Table I
EXPLANATIONS OF THE USED SYMBOLS

Symbol	Parameter
f_i	Node i's physical frequency
θ_i	Node i's physical offset
$\alpha_i^{(\ell)}$	Node i's control parameter α_i at the beginning of ℓth SR
$\beta_i^{(\ell)}$	Node i's control parameter β_i at the beginning of ℓth SR
$\hat{f}_i^{(\ell)}$	Node i's logical frequency at the beginning of ℓth SR
$\hat{\theta}_i^{(\ell)}$	Node i's logical offset at the beginning of ℓth SR

Suppose node i receives a timing message at the time $t_\ell+\delta_\ell$. Here, t_ℓ is the perfect time when the message was sent during the ℓth SR and δ_ℓ is the transmission delay, which includes the PHY layer delay and the propagation delay. We denote the transmitting node by $\tilde{j}(\ell)$. Hence, the received time stamp at the time $t_\ell+\delta_\ell$ is $C_{\tilde{j}(\ell)}(t_\ell)$. For simplicity of presentation and analysis, we ignore δ_ℓ for the time being. However, the final algorithm will be adjusted to be robust against non-negligible transmission delays. Note that, for simplicity of presentation and analysis, we assume $\delta_\ell = 0$ for the time being. However, the final algorithm will be adjusted to be robust against non-negligible transmission delays.

Next we present the two update rules.

1) Partial Update Rule: It is clear that node i cannot make any meaningful update of α_i before receiving at least two timing messages from the same node, since $\hat{f}_i$ basically represents the slope of the linear logical clock model. The aim of the partial update rule at time t_ℓ is to achieve the perfect update (14) when $\hat{f}_{\tilde{j}(\ell)} = \hat{f}_i^{(\ell)}$. That is

$$\begin{cases} \alpha_i^{(\ell+1)} = \alpha_i^{(\ell)} & (17) \\ \beta_i^{(\ell+1)} = \beta_i^{(\ell)} + \frac{1}{2}\left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell)\right), & (18) \end{cases}$$

which implies that

$$\begin{aligned} \hat{\theta}_i^{(\ell+1)} &= \alpha_i^{(\ell+1)}\theta_i + \beta_i^{(\ell+1)} && (19)\\ &= \alpha_i^{(\ell)}\theta_i + \beta_i^{(\ell)} + \frac{1}{2}\left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell)\right) && (20)\\ &= \hat{\theta}_i^{(\ell)} + \frac{1}{2}\left(\hat{f}_{\tilde{j}(\ell)}t_\ell + \hat{\theta}_{\tilde{j}(\ell)} - \hat{f}_i^{(\ell)}t_\ell - \hat{\theta}_i^{(\ell)}\right) && (21)\\ &= \frac{1}{2}\left(\hat{\theta}_{\tilde{j}(\ell)} + \hat{\theta}_i^{(\ell)}\right) + \underbrace{\frac{1}{2}\left(\hat{f}_{\tilde{j}(\ell)} - \hat{f}_i^{(\ell)}\right)t_\ell}_{\triangleq r_i^{(\ell)}}. && (22) \end{aligned}$$

Note that the rest term $r_i^{(\ell)}$ equals to 0 only when $\hat{f}_{\tilde{j}(\ell)} = \hat{f}_i^{(\ell)}$. Nevertheless, we will later prove that, under some conditions, the impact of the rest terms will diminish as $\ell \to +\infty$. For convenience, we define $\Delta_i^{(\ell)}$ as

$$\Delta_i^{(\ell)} \triangleq \frac{1}{2}\left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell)\right), \tag{23}$$

which captures the logical clock adjustment of node i.

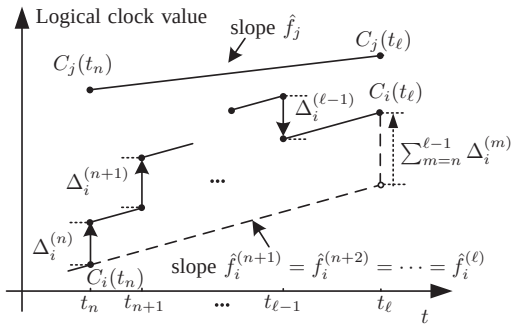


Figure 1. Evolution of a logical clock.

2) Complete Update Rule: Suppose node i receives a timestamp from node $j=\tilde{j}(\ell)$ at time t_ℓ. Furthermore, suppose the last time node i received a timestamp from node j was t_n. Hence, $t_n<t_\ell$ and $j=\tilde{j}(\ell)=\tilde{j}(n)$. Node i will perform a complete update if the following two conditions are satisfied.

a) Node j has not performed a partial or complete update in the interval $(t_n,t_\ell]$.

b) Node i has not performed a complete update in the interval $(t_n,t_\ell]$.

Note that the condition *a)* implies that $C_j(t)$ is an affine function of t, with slope $\hat{f}_j$, for $t\in(t_n,t_\ell]$. For notational simplicity, here we use $\hat{f}_j$ and $\hat{\theta}_j$ without the explicit dependence on the SR index. The condition *b)* implies that $C_i(t)$ is a piecewise affine function of t, with slope $\hat{f}_i^{(n+1)}=\hat{f}_i^{(n+2)}=\cdots=\hat{f}_i^{(\ell)}$, for $t\in(t_n,t_\ell]$. The situation is depicted in Fig. 1. From Fig. 1, it is clear that

$$\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}=\frac{C_j(t_\ell)-C_j(t_n)}{C_i(t_\ell)-\sum_{m=n}^{l-1}\Delta_i^{(m)}-C_i(t_n)}, \tag{24}$$

which is a quantity that will be very useful in the complete update rule. Recall from (13) that we would like to achieve the update $\hat{f}_i^{(\ell+1)}=\left(\hat{f}_j+\hat{f}_i^{(\ell)}\right)/2$ by adjusting $\alpha_i^{(\ell)}$. Therefore, using (5),

$$\frac{1}{2}\left(\hat{f}_j+\hat{f}_i^{(\ell)}\right)=\hat{f}_i^{(\ell+1)}=\alpha_i^{(\ell+1)}f_i \tag{25}$$

$$=\frac{1}{2}\hat{f}_i^{(\ell)}\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right) \tag{26}$$

$$=\frac{1}{2}\alpha_i^{(\ell)}f_i\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right). \tag{27}$$

Correspondingly, the update of α is

$$\alpha_i^{(\ell+1)}=\frac{1}{2}\alpha_i^{(\ell)}\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right), \tag{28}$$

where the right hand side (RHS) can be computed with the help of (24).

Using (6), the desired update for $\hat{\theta}_i$ is according to (14),

$$\frac{1}{2}\left(\hat{\theta}_j+\hat{\theta}_i^{(\ell)}\right)=\hat{\theta}_i^{(\ell+1)}=\alpha_i^{(\ell+1)}\theta_i+\beta_i^{(\ell+1)} \tag{29}$$

$$=\alpha_i^{(\ell+1)}\left(\frac{\hat{\theta}_i^{(\ell)}-\beta_i^{(\ell)}}{\alpha_i^{(\ell)}}\right)+\beta_i^{(\ell+1)}. \tag{30}$$

Therefore, due to (30),

$$\begin{aligned}\beta_i^{(\ell+1)}&=\frac{1}{2}\left(\hat{\theta}_j+\hat{\theta}_i^{(\ell)}\right)-\frac{\alpha_i^{(\ell+1)}}{\alpha_i^{(\ell)}}\hat{\theta}_i^{(\ell)}+\frac{\alpha_i^{(\ell+1)}}{\alpha_i^{(\ell)}}\beta_i^{(\ell)}\\ &\overset{(a)}{=}\frac{1}{2}\left(\hat{\theta}_j-\hat{\theta}_i^{(\ell)}\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right)+\frac{1}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right)\beta_i^{(\ell)}\\ &\overset{(b)}{=}\frac{C_j(t_\ell)}{2}-\frac{\hat{f}_j}{2\hat{f}_i^{(\ell)}}\left(\hat{f}_i^{(\ell)}t_\ell+\hat{\theta}_i^{(\ell)}\right)+\frac{\beta_i^{(\ell)}}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right)\\ &\overset{(c)}{=}\frac{1}{2}\left(C_j(t_\ell)-\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}C_i(t_\ell)\right)+\frac{1}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}}\right)\beta_i^{(\ell)}\end{aligned} \tag{31}$$

where (a) holds by using (28), (b) and (c) follow because of (4). Again, the RHS can be computed with the help of (24).

Remark 5. Even though the MAC layer time stamping can reduce the transmission delays to a large extent, there will still be some delays remaining. When these remaining delays are also taken into account, we can consider a threshold for the adjustment to reduce the influence of delays. When node i receives a timing message at the ℓth SR, it will not use it for adjusting its own logical clock unless $|C_i(t_\ell+\delta_\ell)-C_{\tilde{j}(\ell)}(t_\ell)|>\sigma$, where σ is the threshold. The value of σ is the tradeoff between the speed of the synchronization error decrease and the robustness against delays.

B. Procedures of the RBDS Scheme

Let S_i be the change counter for node i, which is incremented every time node i updates its logical clock (through a partial or complete update).

In the following, let us consider the updates of the logical clock of node i. If node i receives a timing message at the ℓth SR, we assume this message comes from node $\tilde{j}(\ell)$. Hence, the timing message is

$$[\tilde{j}(\ell),\ S_{\tilde{j}(\ell)},\ C_{\tilde{j}(\ell)}(t_\ell)], \tag{32}$$

where t_ℓ is the transmission time, i.e., $t_\ell=\ell\varepsilon+\tau_{\tilde{j}(\ell),\ell}$, where ε is the period of a SR, and $\tau_{j,\ell}$ is the random backoff time drawn by node j at the ℓth SR. Then node i receives this timing message after some delay δ_ℓ and samples its logical clock at time $t_\ell+\delta_\ell$ as $C_i(t_\ell+\delta_\ell)$.

To summarize, node i should update its logical clock given a sequence of the received timing messages in the form of (32) and the corresponding samples of its own logical clock $C_i(t_\ell+\delta_\ell)$ for $\ell=1,2,....$ To keep track of the history, node i maintains a matrix $\boldsymbol{A}_i$, whose rows are of the form

$$[\tilde{j}(\ell),\ S_{\tilde{j}(\ell)},\ C_{\tilde{j}(\ell)}(t_\ell),\ C_i(t_\ell+\delta_\ell)]. \tag{33}$$

Here, the first three elements make up the received timing message at the ℓth SR, and the fourth element is the corresponding sample of the logical clock for node i.

A flow diagram of the proposed scheme is described in Fig. 2, and uses the three algorithms defined below.

[2]The correct receiving means that the message is received without collision. On the other hand, we assume that any collision will lead to packet loss.

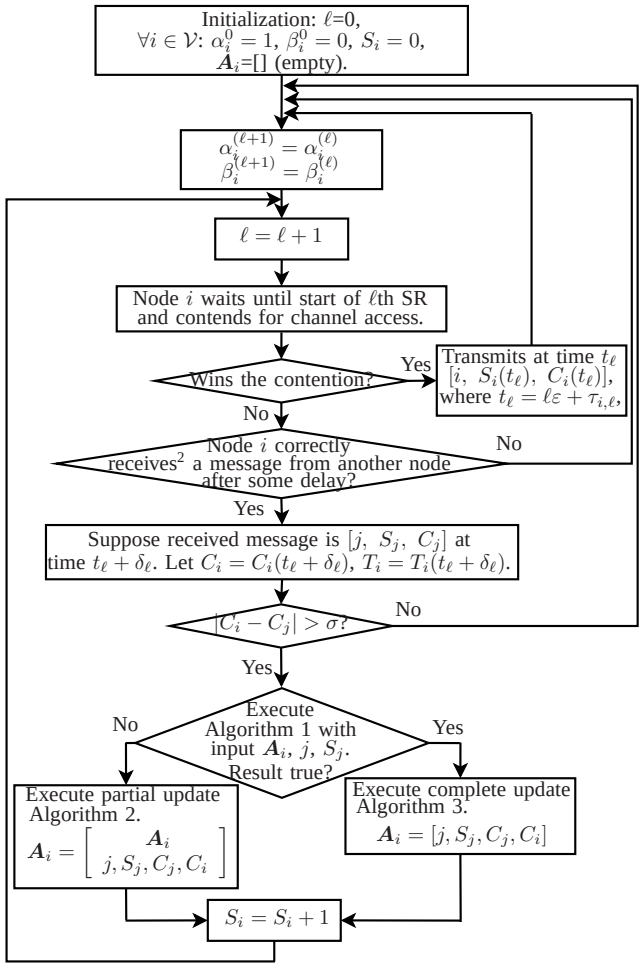


Figure 2. Flowchart of the RBDS scheme.

Algorithm 1 Test if conditions for a complete update are satisfied

Input: $\boldsymbol{A}_i$, j, S_j
Output: true, false
$\mathcal{A} = \{m : [\boldsymbol{A}_i]_{m1} = j\}$ // *the set of previous messages from node j*
if $\mathcal{A} == \emptyset$ **then**
  return false // *no previous message from node j*
else
  $m = \max\{n :\ n \in \mathcal{A}\}$
  if $S_j > [\boldsymbol{A}_i]_{m2}$ **then**
    return false // *node j has changed its logical clock since last timing message*
  else
    return true // *complete update possible*
  end if
end if

C. Convergence of the RBDS Scheme

As revealed in (24), the adjustment of the logical frequency requires at least two timestamps, which cannot be implied by the graph $\mathcal{G}(\ell)$. Therefore, we consider a new directed graph $\mathcal{F}(\ell) = (\mathcal{V}, \tilde{\mathcal{E}}(\ell))$. Regarding the edge set $\tilde{\mathcal{E}}(\ell)$, we define $(i,j) \in \tilde{\mathcal{E}}(\ell)$ if node i implements a complete update at the ℓth SR based on node j's information. Also, $\forall i \in \mathcal{V}$, it is assumed $(i,i) \in \tilde{\mathcal{E}}(\ell)$ for all ℓ. Besides, the set $\tilde{\mathcal{V}}_i(\ell)$ is defined as $\tilde{\mathcal{V}}_i(\ell) = \{j | (i,j) \in \tilde{\mathcal{E}}(\ell)\}$, and $|\tilde{\mathcal{V}}_i(\ell)|$ indicates its cardinality.

Algorithm 2 Partial Update

Input:C_j, C_i, $\alpha_i^{(\ell)}$, $\beta_i^{(\ell)}$
Output: $\alpha_i^{(\ell+1)}$, $\beta_i^{(\ell+1)}$
$\alpha_i^{(\ell+1)} = \alpha_i^{(\ell)}$ // *from (17)*
$\beta_i^{(\ell+1)} = \beta_i^{(\ell)} + (C_j - C_i)/2$ // *from (18)*

Algorithm 3 Complete Update

Input:$\boldsymbol{A}_i$, j, C_j, C_i, $\alpha_i^{(\ell)}$, $\beta_i^{(\ell)}$
Output: $\alpha_i^{(\ell+1)}$, $\beta_i^{(\ell+1)}$
$\rho =$ number of rows in $\boldsymbol{A}_i$
$m = \max\{n : [\boldsymbol{A}_i]_{n1} = j\}$ // *index of the last message from node j*
$\Delta_{\text{tot}} = \sum_{n=m}^{\rho} ([\boldsymbol{A}_i]_{n3} - [\boldsymbol{A}_i]_{n4})$
$\kappa = (C_j - [\boldsymbol{A}_i]_{m3})/(C_i - [\boldsymbol{A}_i]_{m4} - \Delta_{\text{tot}})$ // $\kappa = \hat{f}_j/\hat{f}_i^{(\ell)}$ *from (24)*
$\alpha_i^{(\ell+1)} = \alpha_i^{(\ell)}(1+\kappa)/2$ // *from (28)*
$\beta_i^{(\ell+1)} = (C_j - \kappa C_i)/2 + \beta_i^{(\ell)}(1+\kappa)/2$ // *from (31)*

Besides, the set $\tilde{\mathcal{V}}_i(\ell)$ is defined as $\tilde{\mathcal{V}}_i(\ell) = \{j | (i,j) \in \tilde{\mathcal{E}}(\ell)\}$, and $|\tilde{\mathcal{V}}_i(\ell)|$ indicates its cardinality.

Firstly, according to Theorem 1, the convergence is obvious if the average update rules (13) and (14) can be achieved in the synchronization scheme. However, as analyzed in the two difficulties in Section III-A, it is not straightforward to efficiently implement these average operations. Here, we will propose a theorem to discuss that if (13) and (14) can be attained in the RBDS algorithm.

Theorem 2. *Assume*

a) all nodes can broadcast in any order as long as an infinite sequence of graphs $\mathcal{F}(1)$, $\mathcal{F}(2)$, ... is repeatedly jointly rooted by subsequences of length q;[3]

b) the transmission delays are negligible, i.e., the transmitter and receivers record the timestamps from their local logical clocks simultaneously.

Then, if each node updates its control parameters as (17) and (18), or (28) and (31), depending on whether the conditions of partial update rule or the conditions of complete update rule are satisfied, the average of logical frequencies, i.e., Eq. (13), can be achieved; however, the average of logical offsets, i.e., Eq. (14), is only achieved when $\hat{f}_i = \hat{f}_i^{(\ell)}$.

Proof: Note that the logical frequency will only be modified in the complete update. Suppose node i receive timing messages from node j. By using (28) and $\hat{f}_i^{(\ell)} = \alpha_i^{(\ell)} f_i$, we obtain

$$\hat{f}_i^{(\ell+1)} = \alpha_i^{(\ell+1)} f_i = \frac{1}{2}\left(\hat{f}_j + \hat{f}_i^{(\ell)}\right), \tag{34}$$

which satisfies the average update (13).

[3]This assumption is realistic when a contention based transmission mechanism is used. Moreover, the repeatedly jointly rooted property of $\mathcal{F}(1)$, $\mathcal{F}(2)$, ... implies the repeatedly jointly rooted property of $\mathcal{G}(1)$, $\mathcal{G}(2)$, ...

On the other hand, the logical offset will be changed in both complete update and partial update. In the complete update, by the definition of $\beta_i^{(\ell+1)}$ in (29), we can calculate

$$\hat{\theta}_i^{(\ell+1)} = \alpha_i^{(\ell+1)}\theta_i + \beta_i^{(\ell+1)} = \frac{1}{2}\left(\hat{\theta}_j + \hat{\theta}_i^{(\ell)}\right), \quad (35)$$

which satisfies the average update (14). In the partial update, however, when substituting (17) and (18) into (6), we will obtain

$$\hat{\theta}_i^{(\ell+1)} = \frac{1}{2}\left(\hat{\theta}_j + \hat{\theta}_i^{(\ell)}\right) + \frac{1}{2}\left(\hat{f}_j - \hat{f}_i^{(\ell)}\right) t_\ell, \quad (36)$$

which is not consistent with (14) as long as $\hat{f}_j \neq \hat{f}_i^{(\ell)}$. Therefore, when we have mixed partial and complete updates during certain time period, the average update (14) of the logical offset cannot be attained in general. ■

Even though Theorem 2 proves that (14) is not satisfied in general, the following theorem states that the proposed scheme will achieve consensus asymptotically.

Theorem 3. *Consider the same assumptions a) and b) as in Theorem 2. Then, if each node updates its control parameters as (17) and (18), or (28) and (31), depending on the conditions of the partial update rule or the complete update rule being satisfied, the asymptotical consensus (9) and (10) is achieved, and therefore, (7) is achieved as well.*

Proof: See Appendix A. ■

Remark 6. The proofs of the existing Theorem 1, and our proposed Theorem 2 and Theorem 3 rely on the condition that communication graphs are "repeatedly jointly rooted", an assumption which has also been used in some other works [20], [21]. By introducing a probabilistic framework, the authors in [22] prove that this condition can in fact be satisfied with large probability for any nonzero transmission range and nonzero motion speed, whenever the number of nodes is large enough. Interested readers can find more details in [22, Theorem. 1].

IV. Simulation Results

In this section, simulation results are presented to compare the performance of the proposed RBDS scheme with the following three baseline methods.

1) ATS (with $\rho_\eta = 0.2$, $\rho_o = 0.2$ and $\rho_v = 0.2$) in [8].

ATS includes the cascade of two consensus algorithms where the first consensus synchronizes clock frequencies and the second consensus synchronizes clock offsets. In the first step, nodes broadcast their current estimates of the virtual consensus clock frequency; receiving nodes combine this with their local information to adjust their own virtual consensus clock estimates. The same idea is then applied in the second step to synchronize offsets.

2) MASP in [6].

Based on converge-to-max principle, MASP gives a faster node, which has larger logical clock value, a higher priority to send its synchronization messages. Besides, each node has a self correction capability to compensate the clock oscillation difference among nodes. Finally, slower nodes can synchronize to the fastest node by periodically correcting its clock.

3) CoSyn in [14].

CoSyn is also a random broadcast based distributed synchronization algorithm which can achieve the consensus of both logical clock frequencies and offsets. However, it has different conditions and rules for implementing partial and complete updates compared with RBDS.

We consider a network with N mobile nodes, where nodes are randomly placed in a square-shaped region. The size of the area is 1000 m $\times$ 1000 m. Unless otherwise specified, every node has a fixed transmission range of 250 m which is the same as in [6]. Moreover, all nodes move according to the random way-point model [23] with maximum speed of 40 m/s and 0 pause time. The physical clock frequencies are uniformly and randomly selected from the range $[0.9999, 1.0001]$ Hz, following IEEE 802.11 protocol requirements [3]. Also, the initial clock values are uniformly and randomly chosen from the range $[-800, 800]$ μs. We would like to mention that we have also investigated a wider range of initial clock values, but since the performance of those cases follows the same trends as the ones shown below, we have not included those results, to save space. Besides, as defined in [3], the period of one SR is 0.1 s, and the backoff time in the contention based protocol is uniformly distributed in the range $[0, 1500]$ μs, where the range is calculated with aSlotTime $= 50$ μs and aCWmin $= 15$. Furthermore, as explained above, there still remains some delays even though the MAC layer time stamping is applied. In our simulations, these delays are modeled by a uniform distribution within the range $[0, 2d]$ μs, and we set the threshold $\sigma = d$.

To evaluate synchronization algorithms, we first define the synchronization error between a pair of nodes for each time instance as

$$e_{ij}(t) \triangleq |C_i(t) - C_j(t)|, \forall i, j \in \mathcal{V}, \text{and } j \neq i. \quad (37)$$

Then, based on $e_{ij}(t)$, we adopt two performance metrics in the simulations.

- 90th percentile of synchronization errors, i.e.,

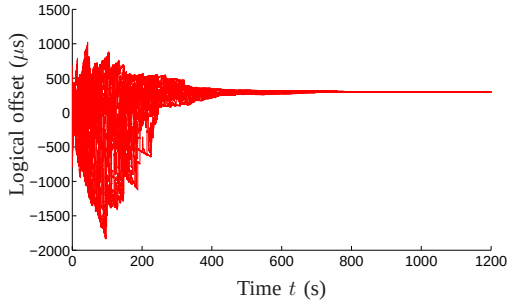
$$e_{90\%}(t) \triangleq x, \quad \text{s.t.} \quad \Pr\{e_{ij}(t) \leq x\} = 90\%. \quad (38)$$

- Probability of unsynchronization with a given threshold γ, i.e.,

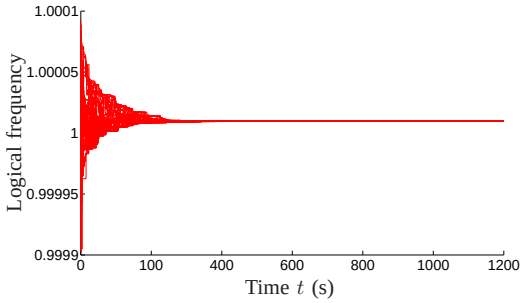
$$P_\gamma(t) \triangleq \Pr\{e_{ij}(t) \geq \gamma\}. \quad (39)$$

For each curve in Fig. 5 to Fig. 10, we average the results over 1000 different network realizations to obtain $\bar{e}_{90\%}(t)$ and $\bar{P}_\gamma(t)$.

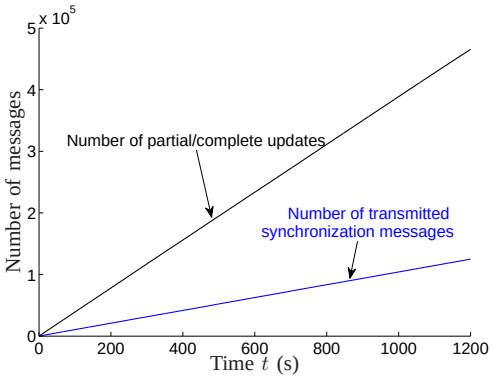
Fig. 3(a) illustrates the convergence of the logical offsets $\hat{\theta}_i$ for $i = 1, \ldots, N$, and Fig. 3(b) shows logical frequencies $\hat{f}_i$ for the proposed RBDS scheme in the absence of delays. It is shown that both logical offsets and logical frequencies will converge to a common value, respectively, which supports Theorem 3 from a numerical perspective. Moreover, convergence can also be evaluated with respect to the number of transmitted synchronization messages as well as the number of partial/complete updates by using the corresponding relation with time in Fig. 3(c). Due to the broadcast nature, the number of partial/complete updates is usually higher than the number of transmitted messages.



(a)



(b)



(c)

Figure 3. Convergence evaluation of logical offsets (Fig. 3(a)), logical frequencies (Fig. 3(b)), and the number of transmitted synchronization messages as well as the number of partial/complete updates (Fig. 3(c)) in the proposed RBDS scheme for 50 nodes, with $d = 0$ (i.e., no delay).

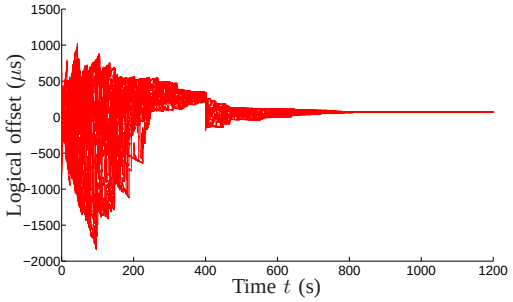
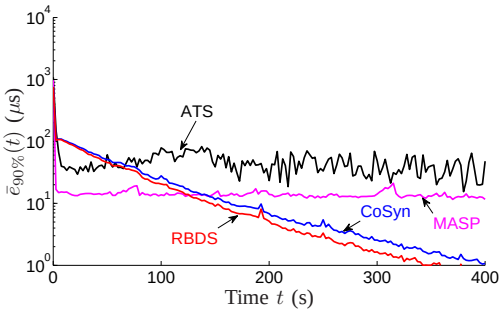
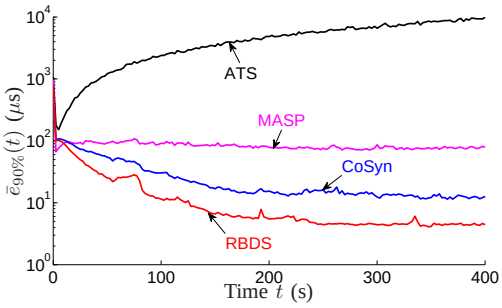


Figure 4. Impact of a frequency variation at 400 s on convergence for 50 nodes, with $d = 0$.



(a)



(b)

Figure 5. 90th percentile of synchronization errors versus time evolution with $N = 50$. (a) $d = 0$; (b) $d = 3$ μs.

The impact of frequency variation on the convergence of the RBDS scheme is evaluated in Fig. 4. Inspired by [24], we assume the physical clock frequencies of half of the nodes are changed at 400 s due to a temperature change. We also consider a typical temperature coefficient -0.04 ppm/°C for crystal oscillators and a temperature change of -25°C. As observed from Fig. 4, even though the logical offsets experience a sudden spread after 400 s due to the frequency variation, the proposed RBDS scheme is able to quickly recover and achieve convergence.

Fig. 5 shows the 90th percentile of synchronization errors versus time when $N = 50$. In the absence of delay (Fig. 5(a)), even though ATS and MASP exhibit fast decrease of synchronization errors in the first 150 s, there are floor effects for both of them. These are because of the problem of over-adjusted logical frequencies in ATS, as well as the contradiction between the fastest node asynchronism and the time partitioning in MASP. On the other hand, CoSyn and RBDS present similar trends regarding the decrease of synchronization errors. Nevertheless, RBDS shows faster convergence due to its increased opportunities of complete updates. Furthermore, the effects of delays are considered in Fig. 5(b). It is depicted that the error of ATS is boosted with time, which implies that ATS becomes ineffective under the scenario with delays. Also, the error decrease is very small in MASP. When it comes to CoSyn and RBDS, they both show robustness against delays, where RBDS outperforms CoSyn.

To illustrate the scalability of synchronization schemes, Fig. 6 shows the 90th percentile of synchronization errors versus the number of nodes N at $t = 400s$. In general, as N increases, so does the network connectivity and the message

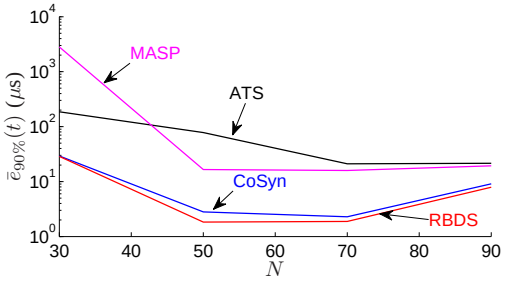


Figure 6. 90th percentile of synchronization errors versus N at $t = 400s$, with $d = 0$.

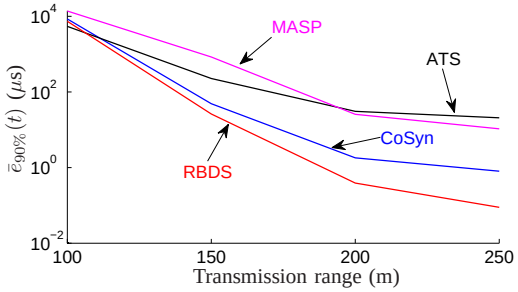


Figure 7. 90th percentile of synchronization errors versus transmission range at $t = 800s$, with $N = 50$ and $d = 0$.

collision rate. The first effect is beneficial for convergence, while the latter is harmful. As seen in Fig. 6, increasing N from $N = 30$ is initially beneficial, but after some point, the collision rate effect will start to dominate and convergence performance will decrease as N is further increased. Regarding the comparison among different synchronization algorithms, CoSyn and RBDS reveal better scalability compared to ATS and MASP, where RBDS slightly outperforms CoSyn.

The influence of transmission range, which varies from 100 m to 250 m, is evaluated in Fig. 7. In general, decreasing the transmission range leads to slower convergence due to reduced network connectivity. When the transmission range is quite small, e.g., 100 m, ATS shows superior performance to other three schemes. Nevertheless, with increased transmission range, its performance improvement is limited. On the other hand, according to our experience, the proposed RBDS algorithm outperforms the other considered algorithms when transmission range is larger than 120 m.

Fig. 8 shows the 90th percentile of synchronization errors versus the delay level d. The error of ATS is boosted with increasing d, which again reveals its sensibility to delays. Moreover, MASP exhibits a moderate increase of the synchronization error, but the error is not close to zero even if $d = 0$. Compared to ATS, MASP, and CoSyn, the proposed RBDS has slower error increase with d, which indicates its robustness against different delay levels.

Fig. 9 shows the probability of unsynchronization as a function of time when setting $\gamma = 10$ μs. When $d = 0$ (Fig. 9(a)), after 400 s, we can see the floor effects for ATS and MASP, where the probabilities attain roughly steady states at 0.08 and 0.18 for ATS and MASP, respectively. The probability of unsynchronization for CoSyn and RBDS, on

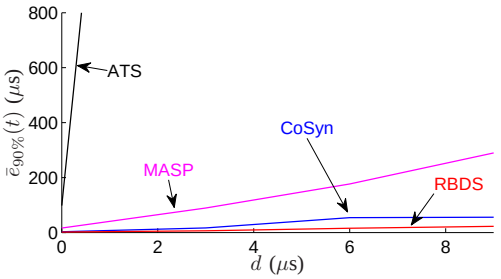


Figure 8. 90th percentile of synchronization errors versus the delay level d at $t = 400s$, with $N = 50$.

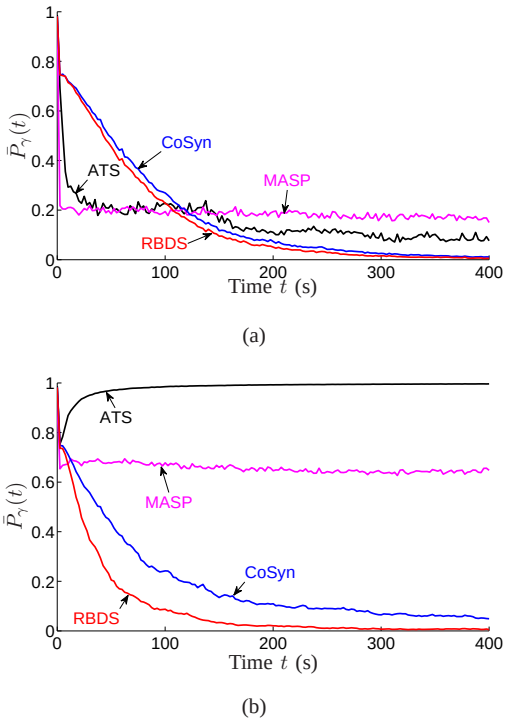


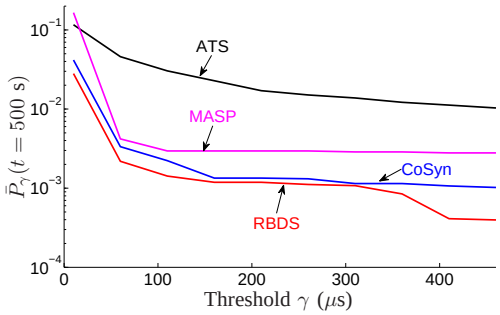
Figure 9. Probability of unsynchronization versus time evolution with $\gamma = 10$ μs and $N = 50$. (a) $d = 0$; (b) $d = 3$ μs.

the other hand, decreases smoothly with time, and RBDS exhibits slightly better performance. After introducing delays, as shown in Fig. 9(b), probability of unsynchronization for ATS increases significantly, which again reveals its sensitiveness to delays. The performance of MASP is also degraded, and its probability of unsynchronization stays around 0.65. While, for both CoSyn and RBDS, the probability of unsynchronization decay with time.

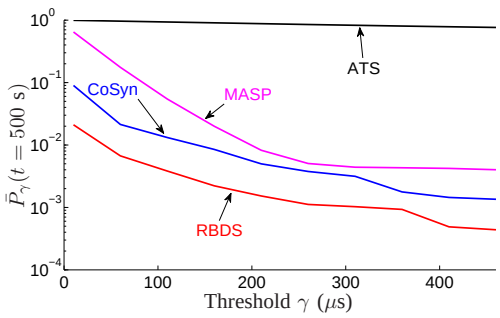
From Fig. 10, we can see how the probability of unsynchronization changes with different thresholds. Here the thresholds vary from 10 μs to 460 μs, which correspond to different accuracy requirements of synchronization. As shown in Fig. 10, in both scenarios of $d = 0$ and $d = 3$ μs, RBDS exhibits superiority over other three algorithms.

V. Conclusions

In this paper, we have proposed a novel and fully distributed consensus clock synchronization scheme—RBDS—for mobile



(a)



(b)

Figure 10. Probability of unsynchronization versus threshold γ at $t = 500$ s, with $N = 50$. (a) $d = 0$; (b) $d = 3$ μs.

ad hoc networks in which timing massages are broadcast in a random manner. In the absence of transmission delays, we theoretically prove the consensus of the logical frequencies and the consensus of the logical offsets for the RBDS approach. Furthermore, as illustrated in the simulations, the RBDS algorithm shows convergence, scalability, and robustness against different transmission delays as well as different accuracy requirements.

Appendix A
Proof of Theorem 3

Firstly, as proved in Theorem 2, the adjustment of the logical frequency follows the average update rule. Therefore, according to Theorem 1, the consensus (9) can be achieved with exponential speed. It is equivalent to

$$\left|[\boldsymbol{r}^{(\ell)}]_i\right| \leq we^{-\lambda\varepsilon\ell}\varepsilon\ell, \quad \forall i \in \mathcal{V}, \tag{40}$$

for some $\lambda > 0$ and $w > 0$, where $\left[\boldsymbol{r}^{(\ell)}\right]_i \triangleq r_i^\ell$ from (22). Note that $\left[\boldsymbol{r}^{(\ell)}\right]_i = 0$ if node i implement the complete update or no update at the ℓth SR.

Then we will move on to the proof of the consensus (10) of the logical offsets. Assume the initial state of the logical offsets of all the N nodes is given by a vector $\boldsymbol{\theta}^{(0)}$. At the ℓth SR, the transfer matrix is defined as $\boldsymbol{P}_\ell$. Note that $\boldsymbol{P}_\ell$ is a stochastic matrix, where a stochastic matrix is defined as a nonnegative square matrix with the property that all its row sums are $+1$. Hence, based on equations (35) and (36), the update of $\boldsymbol{\theta}$ in vector form is given by

$$\boldsymbol{\theta}^{(1)} = \boldsymbol{P}_1\boldsymbol{\theta}^{(0)} + \boldsymbol{r}^{(1)}, \tag{41}$$

$$\boldsymbol{\theta}^{(2)} = \boldsymbol{P}_2\boldsymbol{\theta}^{(1)} + \boldsymbol{r}^{(2)} = \boldsymbol{P}_2\boldsymbol{P}_1\boldsymbol{\theta}^{(0)} + \boldsymbol{P}_2\boldsymbol{r}^{(1)} + \boldsymbol{r}^{(2)}, \tag{42}$$

$$\ldots,$$

$$\boldsymbol{\theta}^{(M)} = \prod_{\ell=1}^{M}\boldsymbol{P}_\ell\boldsymbol{\theta}^{(0)} + \sum_{\ell=1}^{M-1}\prod_{q=\ell+1}^{M}\boldsymbol{P}_q\boldsymbol{r}^{(\ell)} + \boldsymbol{r}^{(M)}. \tag{43}$$

Note that we assume M is an even number here, but it is straightforward to extend the proof for the odd number case.

In order to show the convergence of the logical offsets, it is enough to show the following three items:

a). $\lim_{M\to+\infty}\prod_{\ell=1}^{M}\boldsymbol{P}_k\boldsymbol{\theta}^{(0)} = c\mathbf{1}$, where c is a finite constant;

b). $\lim_{M\to+\infty}\boldsymbol{r}^{(M)} = \mathbf{0}$;

c). $\lim_{M\to+\infty}\sum_{\ell=1}^{M-1}\prod_{q=\ell+1}^{M}\boldsymbol{P}_q\boldsymbol{r}^{(\ell)} = \tilde{c}\mathbf{1}$, where $\tilde{c}$ is a finite constant.

Before the proof of the three items, we firstly re-formulate Theorem 1 in vector form and present several lemmas which will be used later.

Each graph $\mathcal{G}(\ell)$ satisfying the conditions in Theorem 1 can be represented by a transfer matrix $\tilde{\boldsymbol{P}}_\ell$. Note that $\tilde{\boldsymbol{P}}_\ell$ is a stochastic matrix. In this way, the vector form of the update rule (11) is

$$\boldsymbol{x}^{(\ell+1)} = \tilde{\boldsymbol{P}}_k\boldsymbol{x}^{(\ell)}, \tag{44}$$

where $[\tilde{\boldsymbol{P}}_k]_{ij}$ is defined as

$$[\tilde{\boldsymbol{P}}_\ell]_{ij} = \begin{cases} \frac{1}{|\mathcal{V}_i(\ell)|}, & \text{if} \quad j \in \mathcal{V}_i(\ell) \\ 0, & \text{otherwise} \end{cases}. \tag{45}$$

Lemma 1. *If $\boldsymbol{S}_k$ is a stochastic matrix for all $k = 1, 2, ...$, then $\tilde{\boldsymbol{S}}_n \triangleq \prod_{k=1}^{n}\boldsymbol{S}_k$ is still a stochastic matrix for any $n \geq 1$.*

Proof: See [8]. ■

Lemma 2. *If $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, ... is a sequence of stochastic matrices, and the sequence of their associated graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$, ... is repeated jointly rooted by sequences of finite length q, then,*

$$\lim_{n\to+\infty}\prod_{k=1}^{n}\boldsymbol{S}_k = \mathbf{1}\boldsymbol{c}^{\mathsf{T}}, \tag{46}$$

for a finite constant vector $\boldsymbol{c} \triangleq [c_1, c_2, ...c_N]^{\mathsf{T}}$.

Proof:

$$\lim_{n\to+\infty}\prod_{k=1}^{n}\boldsymbol{S}_k = \lim_{n\to+\infty}\prod_{k=1}^{n}\boldsymbol{S}_k\boldsymbol{I} \tag{47}$$

$$= \lim_{n\to+\infty}\left[\prod_{k=1}^{n}\boldsymbol{S}_k\boldsymbol{e}_1, \prod_{k=1}^{n}\boldsymbol{S}_k\boldsymbol{e}_2, ..., \prod_{k=1}^{n}\boldsymbol{S}_k\boldsymbol{e}_N\right] \tag{48}$$

$$= \mathbf{1}\left[c_1, c_2, ...c_N\right] = \mathbf{1}\boldsymbol{c}^{\mathsf{T}} \tag{49}$$

where $\boldsymbol{e}_i$ is the ith column of the identity matrix $\boldsymbol{I}$. The first equality in (49) holds according to Theorem 1. ■

Lemma 3. *If $\boldsymbol{S}$ is a stochastic matrix and $|[\boldsymbol{\phi}]_i| \leq c$, then $|[\boldsymbol{S}\boldsymbol{\phi}]_i| \leq c$ for all $i = 1, 2, ..., N$.*

Proof:

$$|[\boldsymbol{S}\boldsymbol{\phi}]_i| = \left|\sum_{j=1}^{N}[\boldsymbol{S}]_{ij}[\boldsymbol{\phi}]_j\right| \leq \sum_{j=1}^{N}[\boldsymbol{S}]_{ij}\,|[\boldsymbol{\phi}]_i| \tag{50}$$

$$\leq \sum_{j=1}^{N} [\boldsymbol{S}]_{ij} c = c, \tag{51}$$

where the inequality in (50) holds since $\boldsymbol{S}$ is a nonnegative matrix and the equality in (51) holds since $\boldsymbol{S}$ is a stochastic matrix. ∎

Lemma 4. *For all $k = 1, 2, ...$ and $i = 1, ..., N$, if $\boldsymbol{S}_k$ is a stochastic matrix, and $\left|[\boldsymbol{\phi}^{(k)}]_i\right| \leq we^{-\lambda k}k$ for some $\lambda > 0$ and $w > 0$, then, $\left\{a_n \triangleq \left[\sum_{k=1}^{n} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i\right\}$ is a convergent series.*

Proof: By Cauchy's criterion, it suffices to prove $\{a_n\}$ is a Cauchy sequence, i.e., for every $\epsilon > 0$, there is a natural number K, such that for all $n, m > K$ we have that $|a_n - a_m| < \epsilon$ [25].

Without loss of generality, we assume $n > m$ and set $z \triangleq n - m$. So, for all $n > m$,

$$|a_n - a_m| = \left| \left[\sum_{k=1}^{n} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i - \left[\sum_{k=1}^{m} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i \right| \tag{52}$$

$$= \left| \left[\sum_{k=m+1}^{m+z} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i \right| \leq \sum_{k=m+1}^{m+z} \left| \left[\boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i \right| \tag{53}$$

$$\leq \sum_{k=m+1}^{m+z} we^{-\lambda k}k \leq \lim_{z\to+\infty} \sum_{k=m+1}^{m+z} we^{-\lambda k}k \tag{54}$$

$$= \frac{wme^{-\lambda m}}{e^{\lambda} - 1} + \frac{we^{\lambda(1-m)}}{(e^{\lambda} - 1)^2} \triangleq g(m), \tag{55}$$

where the first inequality in (54) follows due to Lemma 3, and the equality in (55) follows from algebraic manipulations. Moreover, it is easy to show that $\lim_{m\to+\infty} g(m) = 0$. Hence, for every $\epsilon > 0$, there exists an M such that $|g(m)| < \epsilon$ whenever $m > M$. Set K as the smallest integer satisfying $K \geq M$. Then, $\forall n > m > K$, we have $|a_n - a_m| \leq g(m) < \epsilon$, which concludes our proof. ∎

Lemma 5. *For all $k = 1, 2, ...$ and $i = 1, ..., N$, if $\boldsymbol{S}_k$ is a stochastic matrix, and $\left|[\boldsymbol{\phi}^{(k)}]_i\right| \leq we^{-\lambda k}k$ for some $\lambda > 0$ and $w > 0$, then,*

$$\lim_{n\to+\infty} \left| \Big[\sum_{k=\frac{n}{2}+1}^{n-1} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)} \Big]_i \right| = 0. \tag{56}$$

Proof:

$$\lim_{n\to+\infty} \left| \Big[\sum_{k=\frac{n}{2}+1}^{n-1} \boldsymbol{S}_k \boldsymbol{\phi}^{(k)} \Big]_i \right| \leq \lim_{n\to+\infty} \sum_{k=\frac{n}{2}+1}^{n-1} \left| \left[\boldsymbol{S}_k \boldsymbol{\phi}^{(k)}\right]_i \right| \tag{57}$$

$$\leq \lim_{n\to+\infty} \sum_{k=\frac{n}{2}+1}^{n-1} we^{-\lambda k}k = 0, \tag{58}$$

where the inequality in (58) holds due to Lemma 3, and the equality in (58) follows from algebraic manipulations. ∎

Now, we are in the position to prove the above three items.

Firstly, consider item *1)*, where each recursion satisfies the update rule (44). Thus, according to Lemma 2, we have

$$\lim_{M\to+\infty} \prod_{\ell=1}^{M} \boldsymbol{P}_\ell \boldsymbol{\theta}^{(0)} = \mathbf{1}\boldsymbol{c}^{\mathsf{T}} \boldsymbol{\theta}^{(0)} = c\mathbf{1}, \tag{59}$$

where $c \triangleq \boldsymbol{c}^{\mathsf{T}} \boldsymbol{\theta}^{(0)}$.

Then, consider item *2)*. Based on the inequality in (40), we can obtain

$$\lim_{M\to+\infty} \left| [\boldsymbol{r}^{(M)}]_i \right| \leq \lim_{M\to+\infty} we^{-\lambda\varepsilon M}\varepsilon M = 0. \tag{60}$$

Finally, consider item *3)*. By Lemma 1, we know that $\boldsymbol{P}'_\ell \triangleq \prod_{q=\ell+1}^{M} \boldsymbol{P}_q$ and $\boldsymbol{P}''_\ell \triangleq \prod_{q=\ell+1}^{\frac{M}{2}} \boldsymbol{P}_q$ are stochastic matrices. In this way, we have

$$\lim_{M\to+\infty} \sum_{\ell=1}^{M-1} \boldsymbol{P}'_\ell \boldsymbol{r}^{(\ell)}$$

$$= \lim_{M\to+\infty} \sum_{\ell=1}^{\frac{M}{2}} \boldsymbol{P}'_\ell \boldsymbol{r}^{(\ell)} + \lim_{M\to+\infty} \sum_{\ell=\frac{M}{2}+1}^{M-1} \boldsymbol{P}'_\ell \boldsymbol{r}^{(\ell)} \tag{61}$$

$$= \lim_{M\to+\infty} \sum_{\ell=1}^{\frac{M}{2}} \boldsymbol{P}'_\ell \boldsymbol{r}^{(\ell)} \tag{62}$$

$$= \lim_{M\to+\infty} \prod_{k=\frac{M}{2}+1}^{M} \boldsymbol{P}_k \sum_{\ell=1}^{\frac{M}{2}} \boldsymbol{P}''_\ell \boldsymbol{r}^{(\ell)} \tag{63}$$

$$= \lim_{M\to+\infty} \prod_{k=\frac{M}{2}+1}^{M} \boldsymbol{P}_k \times \lim_{M\to+\infty} \sum_{\ell=1}^{\frac{M}{2}} \boldsymbol{P}''_\ell \boldsymbol{r}^{(\ell)} \tag{64}$$

$$= \lim_{M\to+\infty} \prod_{k=\frac{M}{2}+1}^{M} \boldsymbol{P}_k \boldsymbol{b} = \mathbf{1}\tilde{\boldsymbol{c}}^{\mathsf{T}} \boldsymbol{b} = \tilde{c}\mathbf{1}, \tag{65}$$

where $\tilde{c} \triangleq \tilde{\boldsymbol{c}}^{\mathsf{T}} \boldsymbol{b}$, the equality in (62) holds according to Lemma 5, the equality in (63) holds by the definitions of $\boldsymbol{P}'_\ell$ and $\boldsymbol{P}''_\ell$, the equality in (64) holds by the limit rule of product [25] combined with Lemma 2 and Lemma 4, the first equality in (65) holds by defining a bounded vector $\boldsymbol{b} \triangleq \lim_{M\to+\infty} \sum_{\ell=1}^{M/2} \boldsymbol{P}''_\ell \boldsymbol{r}^{(\ell)}$, and the second equality in (65) holds according to Lemma 2. The proof of Theorem 3 is concluded by $\tilde{c} \triangleq \tilde{\boldsymbol{c}}^{\mathsf{T}} \boldsymbol{b}$ in (65).

REFERENCES

[1] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. International Conf. on Embedded Networked Sensor Systems*, November 2003, pp. 138–149.

[2] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. International Conf. on Embedded Networked Sensor Systems*, November 2004, pp. 39–49.

[3] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification*, IEEE Std 802.11 Std., 2012.

[4] D. Zhou and T. H. Lai, "A scalable and adaptive clock synchronization protocol for IEEE 802.11-based multihop ad hoc networks," in *Proc. IEEE Mobile Adhoc and Sensor Conf.*, November 2005, pp. 558–565.

[5] ——, "An accurate and scalable clock synchronization protocol for IEEE 802.11-based multihop ad hoc networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1797–1808, December 2007.

[6] H. K. Pande, S. Thapliyal, and L. C. Mangal, "A new clock synchronization algorithm for multi-hop wireless ad hoc networks," in *Proc. IEEE International Conf. on Distributed Computing Systems*, December 2010.

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

[8] 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, September 2011.

[9] M. K. Maggs and S. G. Okeefe, "Consensus clock synchronization for wireless sensor networks," *IEEE Sensors Journal*, vol. 12, pp. 2269–2277, June 2012.

[10] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. IEEE International Conf. on Information Processing in Sensor Networks*, April 2009, pp. 17–48.

[11] Q. Li and D. L. Daniela, "Global clock synchronization in sensor networks," *IEEE Trans. on Computers*, vol. 55, no. 2, pp. 214–226, February 2006.

[12] A. C. Pinho, D. R. Figueiredo, and F. M. G. Franga, "A robust gradient clock synchronization algorithm for wireless sensor networks," in *Proc. IEEE International Conf. on Communication Systems and Networks*, January 2012.

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

[14] W. Sun, M. R. Gholami, E. G. Ström, and F. Brännström, "Distributed clock synchronization with application of D2D communication without infrastructure," in *Proc. IEEE Global Communications Conf. (GLOBECOM) Workshop*, December 2013.

[15] J. So and N. Vaidya, "MTSF: A timing synchronization protocol to support synchronous operations in multihop wireless networks," University of Illinois at Urbana-Champaign, Tech. Rep., January 2004.

[16] W. Sun, E. G. Ström, F. Brännström, and D. Sen, "Long-term clock synchronization in wireless sensor networks with arbitrary delay distributions," in *Proc. IEEE GLOBECOM*, December 2012.

[17] I. Skog and P. Händel, "Synchronization by two-way message exchanges: Cramér-Rao bounds, approximate maximum likelihood, and offshore submarine positioning," *IEEE Trans. Signal Processing*, vol. 58, no. 4, pp. 2351–2362, Apr. 2010.

[18] C. H. Rentel, "Network time synchronization and code-based scheduling for wireless ad hoc networks," Ph.D. dissertation, Carleton University, 2006.

[19] M. Cao, A. S. Morse, and B. D. O. Anderson, "Reaching a consensus in a dynamically changing environment: convergence rates, measurement delays, and asynchronous events," *SIAM Journal on Control and Optimization*, vol. 47, no. 2, pp. 601–623, March 2008.

[20] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[21] M. Cao, D. A. Spielman, and A. S. Morse, "A lower bound on convergence of a distributed network consensus algorithm," in *Proc. IEEE CDC*, December 2005, pp. 2356–2361.

[22] G. Tang and L. Guo, "Convergence of a class of multi-agent systems in probabilistic framework," *J. Syst. Sci. Complex*, vol. 20, no. 2, pp. 173–197, June 2007.

[23] J. P. Sheu, C. M. Chao, and C. W. Sun, "A clock synchronization algorithm for multi-hop wireless ad hoc networks," in *Proc. IEEE International Conf. on Distributed Computing Systems*, March 2004, pp. 574–581.

[24] B. J. Choi, H. Liang, X. Shen, and W. Zhuang, "DCS: Distributed asynchronous clock synchronization in delay tolerant networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 1045–9219, March 2012.

[25] R. Courant, *Differential and Integral Calculus Vol I*, 1961.