Asian Journal of Control, Vol. 00, No. 0, pp. 1–13, Month 2008
Published online in Wiley InterScience (www.interscience.wiley.com) DOI: 10.1002/asjc.0000

An Algorithm for Accurate Distributed Time Synchronization in Mobile Wireless Sensor Networks from Noisy Difference Measurements

Chenda Liao and Prabir Barooah

I. Introduction

Time synchronization is critical for the functionality and performance of wireless sensor networks. For example, in TDMA-based communication schemes, accurate time synchronization ensures that each node communicates with others in the correct time slots without interfering with others. Furthermore, operation on a pre-scheduled sleep-wake cycle for energy conservation also requires a common notion of time among nodes. However, clocks in sensor nodes run at different speeds due to the imperfectness of quartz crystal oscillators, and a tiny difference on the oscillators of two clocks will cause time readings drift apart over time.

In a common clock model, the local time of node u, $\tau_u(t)$ is related to the global time t as

$$\tau_u(t) = \alpha_u t + \beta_u, \tag{1}$$

where the scalar parameters α_u, β_u are called its *skew* (the relative speed of the clock with respect to t) and *offset* (the time reading when $t = 0$), respectively [1]. In practice, skews are time-varying due to temperature change, aging etc. However, it is common to model the skew of a clock as a constant since its variation is negligible during time intervals of interest [2]. A node that knows the global time is called a *reference node*. The global time can be Coordinated Universal Time (UTC) if the reference node(s) can access it through GPS. If none of the nodes can access the UTC, one node in the network is elected as a reference node so that the local time in the node becomes the global time. A node u can determine the global time t from its local time if it knows its skew and offset. Specifically, if u has estimates $\hat{\alpha}_u, \hat{\beta}_u$ of its true skew and offset α_u, β_u, it can estimate the absolute time as

$$\hat{t}_u = \frac{\tau_u(t) - \hat{\beta}_u}{\hat{\alpha}_u} \tag{2}$$

Hence the problem of the time synchronization can be alternatively posed as the problem of skews and offsets estimation among nodes. A node u can use a *pairwise synchronization* method, such as those in [3–5], to estimate α_u and β_u if the paired neighboring node is a reference node. However, most nodes in sensor networks are not connected to the reference nodes directly due to the limited communication range. It is therefore not possible for all nodes to obtain their skews and offsets directly.

Network-wide time synchronization in sensor networks has been intensely studied in recent years. Work in this area can be grouped into three categories: cluster-based protocols, tree-based protocols and distributed protocols. In cluster-based [6] and tree-based protocols [7, 8], synchronization relies on establishing a pre-specified network structure. In mobile networks, however, network topology continually changes, which results in frequent re-computation of a cluster and spanning tree, or re-election of a root node. This introduces considerable communication overhead to the networks, therefore the above cluster-based and tree-based protocols are primarily targeted to networks of static or quasi-static nodes.

Recently, a number of fully distributed algorithms that do not require the establishment of clusters or trees have been proposed. These typically perform synchronization by estimating skews and/or offsets

Manuscript received February 15, 2015.

C. Liao is currently with Nuance Communications,Inc., Burlington, MA 01803 USA (e-mail: cdliao84@gmail.com, ph: 352 870 5251); he was formerly with the Dept. of Mechanical and Aerospace Engineering, University of Florida. P. Barooah is with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: pbarooah@ufl.edu, ph: 352 392 0614)

This work has been supported by the National Science Foundation by Grants CNS-0931885 and ECCS-0955023.

and then computing the global time from them. The algorithms proposed in [9–12] belong to this category. In these algorithms, estimates of a log-skews (and offsets) are obtained from noisy measurements of the difference of log-skews (and offsets) between pairs of neighbors. These distributed algorithms for skew and offset estimation are more readily applicable to mobile networks than the previous two categories of algorithms, though their convergence analyses were provided only for static networks. Convergence of these algorithms in mobile networks is analyzed in [13]. The algorithm analyzed in [13] is applicable to mobile networks and it subsumes the algorithms in [9–12]. The algorithm in [13] is called JaT algorithm (Jacobi-type) due to its similarity to the Jacobi algorithm first proposed in [9]. The time-varying topology of a network of mobile nodes is modeled as the state of a Markov chain. Under certain conditions, it was shown that the variances of estimation errors of log-skews and offsets converge to positive values. However, even a small error in the skew estimate leads to poor absolute time estimate over long time periods, cf. (2). Thus, even a small steady variance of the skew estimates may lead to poor time estimates over time, requiring frequent restarting of the synchronization process.

In this paper, we revisit the problem of distributed estimation of clock skews and offsets from noisy difference measurements. The main contribution is an algorithm (called DiSync) that achieves 0 steady-state variance of the skews and offsets under mild assumptions on the pairwise measurement noise. Mean square convergence of the algorithm is proved for both random (Markovian) as well as deterministic switching of graphs. Time varying gains in the proposed algorithm that make the variances converge to 0 are adopted from stochastic approximation, which is also used in [14, 15] to attenuate noise. The gains need to vary in a specific manner with time, which poses a challenge in implementation in a network of unsynchronized clocks. We address this issue by using a novel approach: an iteration schedule is pre-specified to the nodes so that they can effectively perform a synchronous update without having synchronized clocks. This makes DiSync fully distributed and asynchronous. Furthermore, we propose a DiSync-I algorithm in which the effect of slow convergence rate of the DiSync is ameliorated while retaining theoretical convergence guarantees. We evaluate the accuracy of the DiSync and DiSync-I algorithms when applying to global time estimation through Monte Carlo simulations. Time estimation accuracy of the proposed algorithms are compared with that of the JaT algorithm. Simulations indicate the global time estimation error in the proposed algorithms stay close to 0 for long time intervals, while that in JaT increases over time.

The simulation studies in papers [9–13, 16] use noisy difference measurements generated by adding noise on true values, while in practice these difference measurements are supposed to be obtained from processing multiple time-stamps by using an existing pairwise synchronization protocol, such as the ones proposed in [3–5]. In contrast, here we generate noisy difference measurement by simulating the pairwise synchronization protocol of [3] with random delays in packet reception. The noise in the difference measurements obtained are thus likely to have more realistic characteristics.

A new type of *virtual* time-synchronization protocols has been proposed recently. They let nodes estimate a common virtual global time that is not the local time of any clock [17, 18]. These algorithms are potentially applicable to mobile networks, though not useful when knowing an absolute global time is critical. Still, we compare the proposed algorithm to the ATS algorithm proposed in [17]. Although ATS does not provide estimate of an absolute global time, we compare the algorithms in terms of the maximum synchronization error - the maximum deviation in the estimates of global time (virtual or absolute) over two arbitrary nodes. It turns out that the proposed DiSync and DiSync-I algorithm outperforms ATS under this metric.

The DiSync algorithm bears a close resemblance to average-consensus (leaderless) algorithms in [17, 18]. The estimation error dynamics in our problem turns out to be a leader-follower consensus algorithm, where the leader states - corresponding to the estimation error of the reference nodes - are always 0. The convergence analysis in this paper is inspired from [17, 18]. There are some technical differences since our scenario is leader-follower consensus while those in [17, 18] are leaderless consensus.

A preliminary version of this paper has been published in [19]. Compared to [19] this paper contains several major extensions. First of all, the switching topology was assumed to be deterministic for the convergence analysis in [19], while in this paper we extend the analysis of convergence to Markovian switching. The relevance of Markovian switching comes from the fact that switching topology due to random node motion can be modeled as Markovian [20]. The modified algorithm DiSync-I, which ameliorates the slow convergence rate of theDiSync algorithm, is another novel aspect of

this paper compared to [19]. Moreover, practical implementation details of the algorithm, including extensive simulation comparisons with competing algorithms, is provided here which was lacking in [19].

II. Problem formulation

The time synchronization problem is formulated as nodes estimating their skews and offsets. It is possible for a pair of nodes u, v, who can communicate with each other to estimate their *relative skew* $\alpha_{u,v} := \frac{\alpha_u}{\alpha_v}$ and *relative offset* $\beta_{u,v} := \beta_u - \beta_v \frac{\alpha_u}{\alpha_v}$. The reason for this terminology is the following relationship $\tau_u(t) = \frac{\alpha_u}{\alpha_v}\tau_v(t) + \beta_u - \beta_v \frac{\alpha_u}{\alpha_v}$, that can be derived from (1). The estimation of relative skews and offsets is called "pairwise synchronization". Several protocols for pairwise synchronization from time-stamped messages are available; see [3–5] and references therein. We assume nodes can estimate relative skews and offsets by using one of these existing protocols. Only those nodes that can communicate directly with the reference nodes can estimate their (absolute) skews and offsets, since they can employ pairwise synchronization with reference nodes. Most of the nodes cannot estimate their skews and offsets due to limited communication range.

Suppose between a pair u and v, node u obtains noisy estimates $\hat{\alpha}_{u,v}, \hat{\beta}_{u,v}$ of the parameters $\alpha_{u,v}, \beta_{u,v}$ by using a pairwise synchronization protocol. We model the noisy estimate as $\hat{\alpha}_{u,v} = \alpha_{u,v} + e^s_{u,v}$, where $e^s_{u,v}$ is the estimation error. Therefore, by $\alpha_{u,v} = \frac{\alpha_u}{\alpha_v}$,

$$\log \hat{\alpha}_{u,v} = \log \alpha_u - \log \alpha_v + \xi^s_{u,v}, \tag{3}$$

where $\xi^s_{u,v} = \log(1 + e^s_{u,v}\frac{\alpha_v}{\alpha_u})$. The quantity obtained from pairwise synchronization is therefore a noisy difference measurement of log-skews. If $\alpha_v/\alpha_u \approx 1$, which is usually the case, and $e^s_{u,v}$ is small, then the measurement noise $\xi^s_{u,v}$ is small. Similarly, the noisy estimate of relative offset is modeled as $\hat{\beta}_{u,v} = \beta_{u,v} + e^o_{u,v}$, where $e^o_{u,v}$ is the error. Again, by $\beta_{u,v} = \beta_u - \beta_v \frac{\alpha_u}{\alpha_v}$,

$$\hat{\beta}_{u,v} = \beta_u - \beta_v + \xi^o_{u,v}, \tag{4}$$

which is a noisy difference measurement of the offsets between the two nodes, with measurement noise $\xi^o_{u,v} = \beta_v(1 - \frac{\alpha_u}{\alpha_v}) + e^o_{u,v}$. Due to the nonzero $\beta_v(1 - \frac{\alpha_u}{\alpha_v})$, the measurement error is biased even if $e^o_{u,v}$ is zero mean. Since $\frac{\alpha_u}{\alpha_v}$ is close to 1 for most clocks, the bias is usually small.

We see from (3) and (4) that $\log \hat{\alpha}_{u,v}$ and $\hat{\beta}_{u,v}$ are the noisy measurements of log-skew difference $\log \alpha_u - \log \alpha_v$ and offset difference $\beta_u - \beta_v$, respectively. *The problem of interest for this paper is estimate the log-skews and offsets of all the nodes in a distributed manner from these noisy pairwise difference measurements.* Note that once a node estimates its log-skew, it can recover the skew, and then compute the global time from its local time using estimated skew and offset.

To facilitate further discussion, in this section we only consider the estimation of scalar valued node variables from noisy difference measurements. If an algorithm of solving this problem is available, two copies of the algorithm can be executed in parallel to obtain both log-skews and offsets. Let u-th node in a n-node network have an associated constant scalar *node variable* $x_u \in \mathbb{R}$, $u \in \mathcal{V} = \mathcal{V}_b \cup \mathcal{V}_r = \{1, \ldots, n\}$. Nodes in $\mathcal{V}_b = \{1, \ldots, n_b\}$ do not know their node variables, while the reference nodes are the remaining n_r nodes in $\mathcal{V}_r = \{n_b + 1, \ldots, n\}$, who know the values of their own node variables. Here x_u represents $\log(\alpha_u)$ for skew estimation and β_u for offset estimation. Without loss of generality, we assume node variables of reference nodes are all 0, i.e. skews are 1 and offsets are 0. Time is measured by a discrete time-index $k = 0, 1, \ldots$. The mobile nodes define a time-varying undirected *measurement graph* $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$, where $(u, v) \in \mathcal{E}(k)$ if and only if u and v can obtain a difference measurement of the form

$$\zeta_{u,v}(k) = x_u - x_v + \xi_{u,v}(k), \tag{5}$$

during the time interval between k and $k+1$, where $\xi_{u,v}(k)$ is measurement error. We assume that between u and v, whoever obtains the measurement first shares it with the other so that it is available to both u and v. We also follow the convention that the difference measurement between u and v that is obtained by the node u is always of $x_u - x_v$ while that used by v is always of $x_v - x_u$. Since the same measurement is shared by a pair of neighboring nodes, if v receives the measurement $\zeta_{u,v}(k)$ from u, then it converts the measurement to $\zeta_{v,u}(k)$ by assigning $\zeta_{v,u}(k) := -\zeta_{u,v}(k)$. For similar reasons, between a pair u and v, the node who computes $\zeta_{u,v}(k)$ in node pair u and v is fixed for all time k. This can be achieved by comparing the magnitude of the index of nodes. For example, if $u > v$, then u computes $\zeta_{u,v}(k)$ first and then sends it to v. The *neighbors* of u at k, denoted by $\mathcal{N}_u(k)$, is the set of nodes that u has an edge with in the measurement graph $\mathcal{G}(k)$. We assume that if $v \in \mathcal{N}_u(k)$, then u and v can also exchange information through wireless communication at time k.

Now the reformulated problem is to estimate the node variables x_u, $u \in \mathcal{V}_b$, by using the difference measurements $\zeta_{u,v}(k), (u,v) \in \mathcal{E}(k)$ that become available over time. We assume $n_r \geq 1$ (i.e., there exists a least one reference node), otherwise the problem is indeterminate up to a constant.

III. The DiSync algorithm

The proposed DiSync algorithm is an iterative algorithm that nodes can use to solve the problem of node variable estimation from noisy difference measurements in a distributed and asynchronous manner when they do not have synchronous clocks. Later in the section we describe how this algorithm is used to estimate the log-skews and offsets of the nodes, i.e., perform time-synchronization.

Each node $u \in \mathcal{V}_b$ keeps its local *iteration index* k_u and maintains an estimate $\hat{x}_u(k_u) \in \mathbb{R}$ of its node variable x_u in its local memory. The estimates can be initialized to arbitrary values. In executing the algorithm, node u starts its i-th iteration at a pre-specified local time $\tau^{(i)}$, for $i = 0, 1, \ldots$, which will be described in Section 3.1. Then, node u obtains current estimates $\hat{x}_v(k_u)$ along with the measurements $\zeta_{u,v}(k_u)$ from its current neighbors $v \in \mathcal{N}_u(k_u)$. After a fixed time length δt (measured in local time), node u increments its local iteration index k_u by 1, and updates its new estimate based on current measurements and neighbors' estimates by using the following update law:

$$\begin{aligned}\hat{x}_u(k_u+1) =& \hat{x}_u(k_u) + m(k_u) \sum_{v \in \mathcal{N}_u(k_u)} a_{uv}(k_u)(\hat{x}_v(k_u) \\ &+ \zeta_{u,v}(k_u) - \hat{x}_u(k_u)), \end{aligned} \tag{6}$$

where the time varying gain $m(\cdot) : \mathbf{Z}^+ \to \mathbb{R}^+$ has to be specified to all nodes a-priori. Note that when $\mathcal{N}_u(k_u) = \emptyset$, $\hat{x}_u(k_u+1) = \hat{x}_u(k_u)$. The choice of $m(\cdot)$ will play a crucial role in the convergence of the algorithm and will be described in Section IV. In this paper, we let weight $a_{uv}(k_u) = 1$ if $(u,v) \in \mathcal{E}(k_u)$. Recall that the reference nodes take part by helping their neighbors obtain difference measurements, but they do not update their own node variables. The algorithm is summarized in Algorithm 1. Note that, since obtaining difference measurements requires exchanging time-stamped messages, current estimates can be easily exchanged during the process of obtaining new measurements.

Algorithm 1 DiSync algorithm at node u

```
while u is performing time synchronization do
  if Local time τ_u = τ^(i), i = 0, 1, ... then
    u collects current local indices k_v from neighbors v ∈ N_u(k_u).
    for all v ∈ N_u(k_u) do
      if k_u = k_v and u does not have ζ_{u,v}(k_u) then
        1.u and v perform pairwise communication;
        2.u saves ζ_{u,v}(k_u) and x̂_v(k_u); v saves ζ_{v,u}(k_v) and x̂_u(k_v);
      else
        u and v stop the communication;
      end if
    end for
  end if
  if τ_u = τ^(i) + δt, i = 0, 1, ... then
    if N_u(k_u) ≠ ∅ then
      u updates x̂_u(k_u + 1) using (6);
    else
      x̂_u(k_u + 1) = x̂_u(k_u);
    end if
    u updates, k_u=k_u+1;
  end if
end while
```

3.1. Iteration schedule and synchronous view

We will later describe that the gains $m(\cdot)$ is chosen to be a decreasing function of time, which helps reduce the effect of measurement noise. This is a well-known idea in stochastic approximation. However, using this idea in a network of unsynchronized clocks presents an unique challenge since no node has a notion of a common time index, at least in the initial phase when they do not have good estimates. If nodes wait for a constant length of time (measured in their local clocks) before starting a new iteration, a node with faster skew might finish the $(i+1)$-th iteration while a node with slower skew hasn't even finished the i-th iteration. Therefore, specifying a function $m(\cdot)$ to all the nodes does not ensure that nodes use the same gain at the same (global) interval, which is required by stochastic approximation.

We address the problem by providing the nodes a priori the sequence of local time instants $\tau^{(i)}$, $i = 0, 1 \ldots$ mentioned at the beginning of the Section III. This sequence is called an *iteration schedule*, and the formula for computing it is described below. Let the skews and offsets of all clocks be lower and upper bounded by those in two fictitious clocks c_L and c_H, such that $\alpha_{c_L} \leq \alpha_u \leq \alpha_{c_H}$, $\beta_{c_L} \leq \beta_u \leq \beta_{c_H}$. Recalling (2), therefore $\tau_{c_L}(t) \leq \tau_u(t) \leq \tau_{c_H}(t)$ for all $u \in \mathcal{V}$. The formula for calculating $\tau^{(i)}$ is

$$\tau^{(i+1)} = \frac{\alpha_{c_H}}{\alpha_{c_L}}(\tau^{(i)} + \delta t - \beta_{c_L}) + \beta_{c_H}, \tag{7}$$

where $\tau^{(0)}$ has to satisfy $\tau^{(0)} > \beta_{c_H}$. This schedule ensures that nodes operating on their unsynchronized

local clocks still perform updates in an effectively synchronous manner over time. To see this, define $\mathcal{I}^{(i)} := (\frac{\tau^{(i)}-\beta_{c_H}}{\alpha_{c_H}}, \frac{\tau^{(i+1)}-\beta_{c_H}}{\alpha_{c_H}})$ as a global interval and $\mathcal{I}_u^{(i)} := (\frac{\tau^{(i)}-\beta_u}{\alpha_u}, \frac{\tau^{(i)}+\delta t-\beta_u}{\alpha_u})$ as the global time interval with respect to i-th local iteration of node u. Eq. (7) guarantees that, at each i, $\mathcal{I}_u^{(i)} \subset \mathcal{I}^{(i)}$ for all $u \in \mathcal{V}$. In other words, there exists a sequence of global time intervals such that each i-th global interval contains, and only contains, the i-th local iteration (in global time) of all $u \in \mathcal{V}$. Figure 1(a) shows the relationship between intervals of local iterations and the corresponding global intervals. In Figure 1(b), we pick the 3rd global interval from Figure 1(a), and show the global time intervals when local iteration updates occur. We emphasize that $\tau^{(i)}$ is the same for all nodes and every node u starts and ends its i-th iteration at the same local time instants $\tau^{(i)}$ and $\tau^{(i)} + \delta t$. Each node is provided the values of the parameters $\frac{\alpha_{c_H}}{\alpha_{c_L}}$, β_{c_L},β_{c_H}, and δt ahead of time, which are design variables.

We next address the issue of how to pick values for $\frac{\alpha_{c_H}}{\alpha_{c_L}}$, β_{c_L} and β_{c_H} without knowing a real bounds on skews and offsets of all clocks in a network. In wireless sensor nodes, a pair of clocks in sensor nodes usually drift apart up to 40 $\mu sec/sec$ [2]. Therefore, we can pick $\alpha_{c_H}/\alpha_{c_L} \approx 1 + 4 \times 10^{-5}$. To pick reasonable values of the offset bounds, the following procedure should be used to initialize the synchronization. The reference node first broadcasts a message (to indicate the beginning of synchronization) and sets its local clock time t to zero simultaneously. A node that receives this message sets its own clock to zero and broadcast such message again. The nodes that hear this message also set their local clocks to 0, and so forth. Since nodes (except the reference node) start their local clocks after – but close to – the instant of $t = 0$, their offsets are negative and small. Therefore, β_{c_H} can be chosen as zero and β_{c_L} can be picked as an estimate of the time it takes for all active nodes to receive the "synchronization start" signal. For a node who was out of communication range at the beginning but joins the networks later, it can set the local time to the current local time of a neighbor that has already started the time synchronization, and record neighbor's iteration index as well. In this way, the newly joined node can take part in the synchronization process as if it started at the very beginning.

Another practical issue is the unbounded growth of the inter-synchronization intervals $\tau^{(i)}$. For example, if δt is chosen as 1 second, the choice of $\alpha_{c_H}/\alpha_{c_L} = 1 + 4 \times 10^{-5}$ ensures that the time interval between two successive iterations, $\tau^{(i+1)} - \tau^{(i)}$, will increase from 1 second to 60 seconds after 1.023×10^5 iterations, or 28.4 hours. If the updates are to be done more slowly, a larger δt will be used, which will slow down the growth of the iteration interval $\tau^{(i+1)} - \tau^{(i)}$. If it is desired that the inter-synchronization interval not increase beyond a certain pre-specified bound, a reference node should restart the synchronization after a certain time to maintain $\tau^{(i+1)} - \tau^{(i)}$ within the desired bound. When the restart should occur can be computed from the recursion (7).

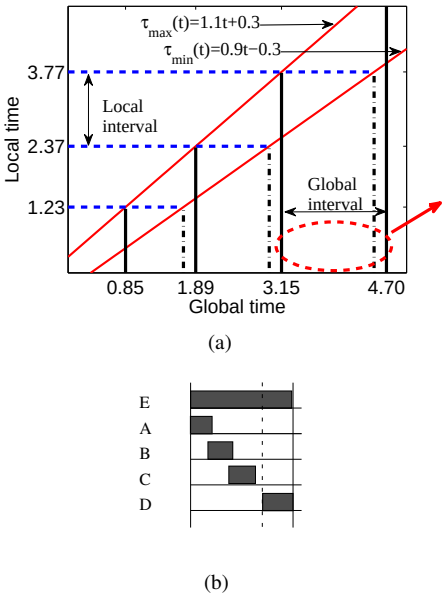


Fig. 1. In (a), the X-axis is labeled by the time instants of the beginning of global intervals and the Y-axis is labeled by the sequence of $\tau^{(i)}$. The red solid slanted lines represent two fictitious clocks c_L and c_H as the bounds for local clocks in nodes. The black solid vertical lines divide global time into a sequence of $\mathcal{I}^{(i)}$. Each i-th interval from black solid vertical to black dotted-dash vertical line is the interval $(\frac{\tau^{(i)}-\beta_{c_H}}{\alpha_{c_H}}, \frac{\tau^{(i)}-\beta_{c_L}}{\alpha_{c_L}})$, which contains the global time instants of $\tau^{(i)}$ for all $u \in \mathcal{V}$. In (b), the 3rd global interval is picked as also circled in (a). The segments in the second and fifth rows correspond to $\mathcal{I}_{c_H}^{(3)}$ and $\mathcal{I}_{c_L}^{(3)}$ of two fictitious clocks respectively. The segments in the third and fourth rows present $\mathcal{I}_u^{(3)}$ and $\mathcal{I}_v^{(3)}$ of any two nodes $u, v \in \mathcal{V}$ accordingly.

Remark 1 *Nodes perform skew and offset estimation simultaneously in a distributed and iterative fashion by using two copies of the DiSync algorithm, one for skew estimation and one for offset estimation. With current estimated skew $\hat{\alpha}_u(k)$ and offset $\hat{\beta}_u(k)$, a node u can compute the global time using (2), i.e. $\hat{t}_u(k) = (\tau_u - \hat{\beta}_u(k))/\hat{\alpha}_u(k)$, which is the final step of the*

Fig. 2. Time synchronization by using DiSync.

time synchronization. The entire time synchronization procedure is illustrated in Figure 2.

IV. Convergence analysis of DiSync algorithm

Since there exists a common iteration counter k that can be used to describe the local updates in the nodes by using the iteration schedule (even though none of the nodes has access to it), we consider only the synchronous version of the algorithm using global index k from now on. We rewrite (6) as

$$\begin{aligned}\hat{x}_u(k+1) =&\hat{x}_u(k) + m(k) \sum_{v\in\mathcal{N}_u(k)} a_{uv}(k)(\hat{x}_v(k) \\ &+ \zeta_{u,v}(k) - \hat{x}_u(k)).\end{aligned} \tag{8}$$

Defining the estimation error as $e_u(k) := \hat{x}_u(k) - x_u$, Eq. (8) reduces to the following using (5):

$$\begin{aligned}e_u(k+1) =&e_u(k) + m(k) \sum_{v\in\mathcal{N}_u(k)} a_{uv}(k)(e_v(k) \\ &- e_u(k) + \epsilon_{u,v}(k)).\end{aligned} \tag{9}$$

We introduce the following stipulations and notations to pursue subsequent analysis. First,let $a_{uv}(k) = 0$ for $v \notin \mathcal{N}_u(k)$. Secondly, the $n \times n$ Laplacian matrix $L(k)$ of the graph $\mathcal{G}(k)$ is defined as $L_{uv}(k) = \sum_{v=1}^{n} a_{uv}(k)$ if $u = v$, and $L_{uv}(k) = -a_{uv}(k)$ if $u \neq v$. By removing the rows and columns of $L(k)$ with respect to reference nodes, we get the $n_b \times n_b$ principle submatrix $L_b(k)$ (so called grounded or Dirichlet Laplacian matrix [21]). Let $e(k) := [e_1(k), \ldots, e_{n_b}(k)]^T$, the corresponding state space form of the estimation error is

$$e(k+1) = (I - m(k)L_b(k))e(k) + m(k)D(k)\epsilon(k), \tag{10}$$

where

$$\begin{aligned}&\epsilon(k) := [\bar{\epsilon}_1(k)^T, \ldots, \bar{\epsilon}_{n_b}(k)^T]^T, \\ &\bar{\epsilon}_u(k) := [\epsilon_{u,1}(k), \ldots \epsilon_{u,n}(k)]^T, \\ &D(k) := diag(\bar{a}_1(k), \ldots, \bar{a}_{n_b}(k)), \\ &\bar{a}_u(k) := [a_{u,1}(k), \ldots, a_{u,n}(k)],\end{aligned}$$

where $\epsilon_{u,u}(k) \notin \bar{\epsilon}_u(k)$ and $a_{u,u}(k) \notin \bar{a}_u(k)$. When $a_{uv}(k) = 0$, $\epsilon_{u,v}(k)$ is a pseudo random variable with the same mean and variance as the measurement noise on any existing edge. Moreover, as a node u computes measurement $\zeta_{u,v}(k)$ and sends $-\zeta_{u,v}(k)$ to v, $\epsilon_{u,v}(k) = -\epsilon_{v,u}(k)$. Now, we introduce the following assumptions:

Assumption 1 *Measurement noise vector $\epsilon(k)$ is with mean $E[\epsilon(k)] = \gamma$ and bounded second moment, i.e. $E[\|\epsilon(k)\|^2] < \infty$, where $\|\cdot\|$ denotes 2-norm. Furthermore, $\epsilon(k)$ and $\epsilon(j)$ are independent for $k \neq j$. In addition, $\{\epsilon(k)\}$ is independent of $e(0)$, where $E\|e(0)\|^2 < \infty$.*

In practice, $E[\epsilon(k)]$ may be time-varying even if all the underlying processes are wide sense stationary. For instance, the bias in offset difference measurement computed from node u is different from that computed from node v, as $\beta_v(1 - \frac{\alpha_u}{\alpha_v}) \neq \beta_u(1 - \frac{\alpha_v}{\alpha_u})$. Therefore, $E[\epsilon(k)]$ depends on which node initializes pairwise synchronization at time k. To meet this requirement $E[\epsilon(k)] = \gamma$ in Assumption 1, we can stipulate that the node who computes $\zeta_{u,v}(k)$ between a pair u and v is fixed for all time k. This can be achieved by comparing the magnitude of the index of nodes. For example, if $u > v$, then u computes $\zeta_{u,v}(k)$ first and then sends it to v. Indeed, the purpose of this requirement is to provide formula to compute the steady state value of estimation error, and the system may still achieve convergence without it.

Assumption 2 *The non-increasing positive sequence $\{m(k)\}$ (step size of the stochastic approximation) is chosen as $m(k) = \frac{c_1}{k+c_2}$, where c_1, c_2 are constant real numbers.*

Note that Assumption 2 is a special case of the standard requirement in stochastic approximation: $\sum_{k=0}^{\infty} m(k) = \infty$ and $\sum_{k=0}^{\infty} m^2(k) < \infty$. The assumption is made to simplify the subsequent analysis, but we believe it is not necessary.

4.1. Deterministic Topology Switching

In this section, we analyze the convergence of (8) when the topology switching is deterministic.

Assumption 3 *There exists $d \in \mathbb{N}$ s.t. for any $t \geq 0, \hat{\mathcal{G}}_t^d := \bigcup_{k=t}^{t+d-1} \mathcal{G}(k) = (\mathcal{V}, \bigcup_{k=t}^{t+d-1} \mathcal{E}(k))$ is connected, where $\mathcal{E}(k)$ is set of edges in $\mathcal{G}(k)$.*

Assumption 4 *The limits* $\bar{L}, \bar{L}_b, \bar{D}$ *exist:* $\bar{L} := \lim_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t} L(k)$, $\bar{L}_b := \lim_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t} L_b(k)$, $\bar{D} := \lim_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t} D(k)$.

Assumption 3 implies that information can go from any node to the rest nodes within a uniformly bounded length of time. Furthermore, as $\mathcal{G}(k)$ is bidirectional, another equivalent assumption is that $\hat{G}_t^d$ contains a spanning tree. The proposed algorithm is also robust to permanently adding or deleting nodes in case the new resulting graph satisfies the assumption on connectivity. To understand the Assumption 4, we define the finite state space $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$ as the set of graphs that can occur over time. If the sequence of $\mathcal{G}(k)$ can be divided into a sequence of finite intervals I_j, $j = 1, 2, \ldots$, such that the percentage of times that each state $\mathcal{G}_k$ occurs is fixed in all except finitely many such intervals I_j, then $\bar{L}$, $\bar{L}_b$ and $\bar{D}$ exist. Another example is that the state $\mathcal{G}_i$ occurs according to a sample path of a stationary ergodic process. In addition, we denote sets of matrices $\mathbb{L}_b = \{L_{b1}, \ldots, L_{bN}\}$ and $\mathbb{D} = \{D_1, \ldots, D_N\}$, where L_{bi} and D_i correspond to $\mathcal{G}_i \in \mathbb{G}$. If the percentage of all states occurring is $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$, then

$$\bar{L}_b := \sum_{i=1}^{N} \pi_i L_{bi}, \ \bar{D} := \sum_{i=1}^{N} \pi_i D_i \tag{11}$$

Theorem 1 *Under Assumption 1-4, the Algorithm 1 ensures that* $e(k)$ *in* (10) *converges to* $\bar{L}_b^{-1}\bar{D}\gamma$ *in mean square, i.e.,* $\lim_{k\to\infty} E(\|e(k) - \bar{L}_b^{-1}\bar{D}\gamma\|^2) = 0$. □

The theorem states that under the assumptions, the variance of the estimation error decays to 0. If additionally all the difference measurements are unbiased ($\gamma = 0$), then the bias of the estimates converge to 0 as well. Proof of the theorem uses the next two lemmas. The proof of the first lemma, which is inspired by [14, 15], can be found in [22], while the second is from[23].

Lemma 1 *If difference measurement is unbiased, i.e.,* $\gamma = 0$, *under assumption 1-3, the Algorithm 1 ensures that* $e(k)$ *in* (10) *converges to* 0 *in mean square, i.e.,* $\lim_{k\to\infty} E(\|e(k)\|^2) = 0$. □

When $\gamma = 0$, (10) can be regarded as a leader-following consensus problem with time-varying topology and zero-mean noisy input. The leaders are reference nodes $u \in \mathcal{V}_r$, which hold their variables as zero. Then, $e_u(k)$ for $u \in \mathcal{V}_b$ is driven to zero by the reference nodes as k goes to ∞ in mean square sense.

Lemma 2 *[23]Denote by* A *an unknown symmetric and positive semi-definite matrix in* $\mathbb{R}^{n\times n}$, *and we have to solve the equation* $Ax=y$ *for an unknown* $y \in \mathbb{R}^n$. *Assume that* A^{-1} *exists. We are given a sequence of matrices* A_k *and a sequence* y_k, *where* $k = 0, 1, \ldots$. *In addition, suppose that* $\lim_{k\to\infty} \|\frac{1}{k}\sum_{i=1}^{k} y_i - y\| = 0$, $\lim_{k\to\infty} \|\frac{1}{k}\sum_{i=1}^{k} A_i - A\| = 0$, $\lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} \|A_i\|^2$ *exists. Consider the sequence* x_k: x_0 *is arbitrary,*

$$x_{k+1} = x_k + \frac{c_1}{k + c_2}(y_k - A_k x_k), \tag{12}$$

where c_1 *and* c_2 *are constant real numbers. Then,* $\lim_{k\to\infty} x_k = A^{-1}y$. □

Proof of Theorem 1 Taking expectations on both sides of (10) with respect to measurement noise $\epsilon(k)$, we obtain

$$\eta(k+1) = (I - m(k)L_b(k))\eta(k) + m(k)D(k)\gamma, \tag{13}$$

where $\eta(k) = E[e(k)]$. By substituting (13) in (10), we get

$$\tilde{e}(k+1) = (I - m(k)L_b(k))\tilde{e}(k) + m(k)D(k)\xi(k), \tag{14}$$

where $\tilde{e}(k) = e(k) - \eta(k)$ and $\xi(k) = \epsilon(k) - \gamma$. Note that $\xi(k)$ is zero mean and satisfies Assumption 1. By Lemma 1, $\tilde{e}(k)$ converges to 0 in mean square. Therefore, $e(k)$ is mean square convergent to $\eta(k)$. Now, we examine the convergence of $\eta(k)$. From the definition of $\bar{L}_b$ and the symmetry of L_{bi}, $\bar{L}_b$ is a symmetric grounded Laplacian of $\hat{\mathcal{G}}$. Since $\hat{G}$ is connected, by Lemma 1 in [21], $\bar{L}_b$ is positive definite. Consequently, $\lambda_m(\bar{L}_b) > 0$. Now, it follows from Lemma 2 that $\lim_{k\to\infty} \eta(k) = \bar{L}_b^{-1}\bar{D}\gamma$. Consequently, $e(k)$ converges to $\bar{L}_b^{-1}\bar{D}\gamma$ in mean square, i.e., $\lim_{k\to\infty} E(\|e(k) - \bar{L}_b^{-1}\bar{D}\gamma\|^2) = 0$.

4.2. Markovian Topology Switching

In this section, we analyze convergence when network topology switches randomly. We model the switching of the network topologies as a Markov chain; the reasonableness of this model for mobile networks has been established in [13].

Assumption 5 *The temporal evolution of the measurement graph* $\mathcal{G}(k)$ *is governed by an N-state homogeneous ergodic Markov chain with state space* $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$, *which is the set of graphs that*

can occur over time. Furthermore $\hat{\mathcal{G}} := \bigcup_{k=1}^{N} \mathcal{G}_k = (\mathcal{V}, \bigcup_{k=1}^{N} \mathcal{E}_k)$ *is connected, where* $\mathcal{E}_k$ *is set of edges in* $\mathcal{G}_k$*. In addition, the processes* $\mathcal{G}(k)$ *and* $\epsilon(j)$ *are independent for all* k *and* j*.*

In Assumption 5, the Markovian switch on the graphs means that $P(\mathcal{G}(k+1) = \mathcal{G}_i | \mathcal{G}(k) = \mathcal{G}_j) = P(\mathcal{G}(k+1) = \mathcal{G}_i | \mathcal{G}(k) = \mathcal{G}_j, \mathcal{G}(k-1) = \mathcal{G}_\ell, \ldots, \mathcal{G}(0) = \mathcal{G}_p)$ where $\mathcal{G}_i, \mathcal{G}_j, \mathcal{G}_\ell, \ldots, \mathcal{G}_p \in \mathbb{G}$. The requirement for ergodicity of the Markov chain ensures that there is an unique steady state distribution with non-zero entries. This means every graph in the state space of the chain occurs infinitely often. Since $\hat{\mathcal{G}}$ is connected, ergodicity implies that information from the reference node(s) will flow to each of the nodes over time. Again, note that none of the graphs that ever occur is required to be a connected graph. Since the Markov chain is ergodic, the steady state distribution of the chain is defined as $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$. Recalling that L_{bi} and D_i correspond to $\mathcal{G}_i \in \mathbb{G}$, we can use the same formula as that in (11) to define $\bar{L}_b$ and $\bar{D}$.

Theorem 2 *Under Assumption 1,2 and 5,* $e(k)$ *in* (10) *is mean square convergent, i.e.,* $\lim_{k\to\infty} E(\|e(k) - E_\epsilon(e(k))\|^2) = 0$*, where* $E_\epsilon(e(k))$ *is expectation of* $e(k)$ *w.r.t. measurement noise* $\epsilon(k)$*, and* $E_\epsilon(e(k))$ *converges to* $\bar{L}_b^{-1}\bar{D}\gamma$ *almost surely.*

The proof of the theorem uses the next two lemmas.

Lemma 3 *If relative measurements are unbiased, i.e.,* $\gamma = 0$*, under 1,2 and 5,* $e(k)$ *in* (10) *converges to* 0 *in mean square, i.e.,* $\lim_{k\to\infty} E(\|e(k)\|^2) = 0$*.*

Lemma 4 *[Proposition 1 in [24]] Assume* $\{A_k, y_k\}$*,* $k = 0, 1, \ldots$ *, is stochastic process on* $(\Sigma, \mathcal{F}, P)$*, where* A_k *is an* $n \times n$ *symmetric positive semidefinite matrix and* y_k *is* $n \times 1$*. Consider the following iteration with arbitrary* x_0*:*

$$x_{k+1} = x_k + \frac{c_1}{k + c_2}(y_k - A_k x_k), \tag{15}$$

where c_1 *and* c_2 *are real constants. If* $A := \lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} E[A_i]$, $y := \lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} E[b_i]$*, and* A *is positive definite, then* $\lim_{k\to\infty} x_k = A^{-1}y$ *almost surely.*

Proof of Lemma 3 is omitted due to its length; it can be found in [22]. Lemma 4 follows in a straightforward manner from the results in [24], so its formal proof is omitted.

Proof of Theorem 2: The proof is similar to that for Theorem 1. Define $\eta(k) = E_\epsilon[e(k)]$, where the expectation is taken with respect to measurement noise $\epsilon(k)$. Take the expectation on both sides of (10), we get

$$\eta(k+1) = (I - m(k)L_b(k))\eta(k) + m(k)D(k)\gamma. \tag{16}$$

By substituting (16) in (10), we get

$$\tilde{e}(k+1) = (I - m(k)L_b(k))\tilde{e}(k) + m(k)D(k)\xi(k), \tag{17}$$

where $\tilde{e}(k) = e(k) - \eta(k)$ and $\xi(k) = \epsilon(k) - \gamma$. Note that $\xi(k)$ is zero mean and satisfies Assumption 1. By Lemma 3, $\tilde{e}(k)$ converges to 0 in mean square. Now, we examine the convergence of $\eta(k)$. We rewrite (16) as

$$\eta(k+1) = \eta(k) + m(k)(D(k)\gamma - L_b(k)\eta(k)). \tag{18}$$

It follows from Assumption 5,

$$\begin{aligned} \lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} E[L_b(k)] &= \sum_{i=1}^{N} \pi_i L_{bi} = \bar{L}_b, \\ \lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} E[D(k)] &= \sum_{i=1}^{N} \pi_i D_i = \bar{D} \end{aligned} \tag{19}$$

From the definition of $\bar{L}_b$ and the symmetry of L_{bi}, $\bar{L}_b$ is a symmetric grounded Laplacian of $\hat{\mathcal{G}}$. Since $\hat{G}$ is connected, by Lemma 1 in [21], $\bar{L}_b$ is positive definite. Consequently, $\lambda_m(\bar{L}_b) > 0$. Furthermore, L_{bi} is positive-semi definite. Then, by Lemma 4 that $\eta(k)$ converges to $\bar{L}_b^{-1}\bar{D}\gamma$ almost surely.

V. Ameliorating slow convergence: DiSync-I

The proposed DiSync algorithm ensures mean square convergence by attenuating the measurement noise using ideas from stochastic approximation. In particular, the gain $m(k)$ that decays slowly with time is instrumental in driving the variances of the estimation error to zero. This slowly decaying gain, however, also makes the convergence rate slow. This is a common feature of stochastic approximation algorithms. We'll see numerical evidence of this in Section VI. In practice, transient performance of the DiSync algorithm can be further improved by modifying the update law, which

we describe next. The modified algorithm is called the DiSync-I(DiSync-Improved) algorithm.

First, we define a scalar state $y_u(k)$ for each node u, which is a surrogate for the distance (number of hops) of node u from the reference nodes. The state $y_u(k)$ is called *average distance of u from the reference nodes* at time k, and is computed by an update law that will be described in Algorithm 2. If u is a refernce node, $y_u(k) = 0$ for all k. To describe the update law of $y_u(k)$ for non-reference nodes, we define the set $\mathcal{S}_u(k) := \{v | y_u(k) \geq y_v(k), v \in \mathcal{N}_u(k)\}$, the subset of neighbors of node u whose average distances are smaller than that of node u at k. The state $y_u(k)$ is updated by averaging all $y_v(k)$ from its neighbors who have smaller average distance; see Algorithm 2. If none of its neighbors have smaller average distance than itself, $y_u(k)$ increases. Both $\mathcal{S}_u(k)$ and $y_u(k)$ are maintained in u at index k.

The node variable update law for the DiSync-I algorithm is given below; where the subscript on the local iteration counter is suppressed:

$$\hat{x}_u(k+1) = \hat{x}_u(k) + h(k) \sum_{v \in \mathcal{H}_u(k)} a_{uv}(k)(\hat{x}_v(k) + \zeta_{u,v}(k) - \hat{x}_u(k)), \quad (20)$$

where

$$h(k) = \begin{cases} \frac{1}{1+|\mathcal{H}_u(k)|} & \text{for} \quad k < k_h \\ m(k-k_h) & \text{for} \quad k \geq k_h, \end{cases} \quad (21)$$

$$\mathcal{H}_u(k) = \begin{cases} \mathcal{S}_u(k) & \text{for} \quad k < k_{\mathcal{H}} \\ \mathcal{N}_u(k) & \text{for} \quad k \geq k_{\mathcal{H}}, \end{cases} \quad (22)$$

where k_h and $k_{\mathcal{H}}$ are constants - that satisfy $k_{\mathcal{H}} \leq k_h$ - that are pre-specified to all nodes.

If $y_u(k) \geq y_v(k)$, it means that node v has been closer to the reference nodes than node u on average (even if node v may be farther from the reference node than u at current index k). This indicates that node v is likely to contain better estimates that u. If u and v are neighbors, u should use the estimate from v to perform update, while v should not use estimates from u.

Note that when $k_h = \infty$ and $k_{\mathcal{H}} = 0$, (20) becomes the JaT algorithm of [13, 16]. As shown in [13, 16], JaT algorithm ensures that the mean of the estimation error converges to a constant (zero if measurement is unbiased) and variance to a constant. In addition, when $k_h = \infty$ and $k_{\mathcal{H}} < \infty$, the resulting update law (20) is a modified version of JaT algorithm: it now uses Algorithm 2 during the initial phase up to $k \leq k_{\mathcal{H}}$. We will call it the JaT-I algorithm.

Table 1 shows how the update law (20) can produce different algorithms depending on the values of the parameters $k_h, k_{\mathcal{H}}$. For simulation studies in this paper on the DiSync-I algorithm, we pick $k_h = k_{\mathcal{H}}$ somewhat arbitrarily. In this case, DiSync-I first adopts JaT-I when $k < k_h$ and then becomes DiSync when $k > k_h$. The reason behind the modification (20) over (8) is the following. First, JaT has better convergence speed during initial phase than that of DiSync. Second, JaT-I has even better convergence rate than JaT due to the use of only those neighbors that have been closer to the reference node(s).

Note that the DiSync-I differs from DiSync only during the initial phase (up to k_h), otherwise it is the same. As a result, the mean square convergence results of DiSync holds for DiSync-I as well.

Algorithm 2 Average distance algorithm at node $u \in \mathcal{V}_b$

```
Initialize y_u(0) = ∞
while u is performing iteration do
    for v ∈ N_u(k) do
        if y_u(k) ≥ y_v(k) and y_v(k) ≠ ∞ then
            S_u(k) ← v;
        end if
    end for
    if S_u(k) ≠ ∅ then
        y_u(k+1) = (Σ_{v∈S_u(k)} y_v(k)) / |S_u(k)|;
    else
        y_u(k+1) = y_u(k) + 0.25;
    end if
    k=k+1;
end while
```

Table 1. Comparison of different algorithms

	$k_{\mathcal{H}} = 0$	$k_{\mathcal{H}} < \infty$	$k_{\mathcal{H}} = \infty$
$k_h = 0$	DiSync	—	—
$k_h < \infty$	—	DiSync-I	—
$k_h = \infty$	JaT	JaT-I	—

VI. Simulation evaluation

We now examine through simulations the time synchronization performance of the DiSync and DiSync-I algorithms, as well as that of JaTand JaT-I algorithms. Finally, they are compared with the virtual time synchronization algorithm ATS [17] in terms of pairwise synchronization error. Simulations are performed in a 10-node mobile network within a $10m \times 10m$ square field. Nodes' motions are generated according to the widely used *random waypoint* (RWP) mobility model [25]. It has been shown in [13] that when nodes move according to the RWP mobility model, the switching of the graphs can be

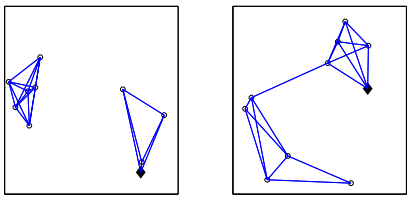


Fig. 3. Two graphs that occur during one simulation with 10 nodes moving according to the random direction mobility model.

modeled as a Markov chain. A pair of nodes can communicate when distance between them is less than $5m$. The true skews and offsets of 9 nodes are picked uniformly from $[1-2\times10^{-5}, 1+2\times10^{-5}]$ and $[-10^{-2}, 10^{-2}]sec$ respectively according to [8]. The single reference node (10th) has skew 1 and offset 0. Denote $k=1,2,\ldots$ as update intervals (also called synchronization periods), and t_k as the global instant of the beginning of k-th interval. In this simulation, the interval is chosen as 1 sec, therefore $t_{k+1}-t_k=1$. For the sake of convenience, simulations are carried out in a synchronous fashion.

6.1. Implementation of pairwise synchronization

The simulation of the DiSync, DiSync-I, JaT and JaT-I algorithms requires pairs of nodes to obtain difference measurements by exchanging time stamped messages when they are within communication range. In order to evaluate the entire time synchronization procedure, unlike [10–12, 16], in which difference measurements are generated by adding random noise to true difference of log-skew/offset, we select the pairwise synchronization algorithm proposed in [3] to compute the relative skew $\alpha_{u,v}$ and relative offset $\beta_{u,v}$, and the difference measurements $\beta_u-\beta_v$ and $\log(\alpha_u)-\log(\alpha_v)$ are then obtained from these as described in Section II. According to [3], at the beginning of the kth interval, node u sends a message to v that contains the value of the local time at u when the message is sent: $\tau_u^{(1)}$. When node v receives this message, it records the local time of reception: $\tau_v^{(1)}$. After a waiting period, node v sends a message back to u that contains both $\tau_v^{(2)}$ and $\tau_v^{(1)}$. When it arrives at u, node u again records the local time of reception: $\tau_u^{(2)}$. Two nodes u and v in communication range performs this procedure, called two-way time-stamped message exchange, twice - at the beginning and in the middle of each synchronization period. At the end of the synchronization period, node u uses the obtained eight time stamps $\{\tau_u^{(i)}, \tau_v^{(i)}\}$ for $i=1,\ldots4$ to estimate $\alpha_{u,v}(k)$ and $\beta_{u,v}(k)$ via the formula provided in [3]. Finally, node u sends back to v these estimates.

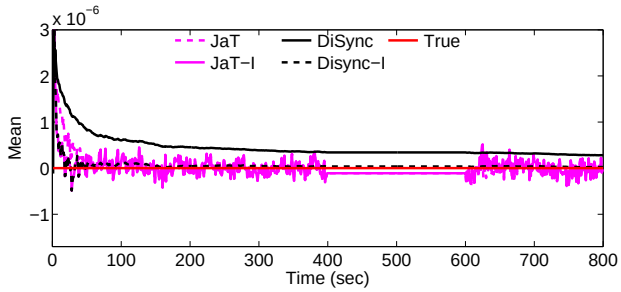
There is a random delay between the time a node sends a message and the other node receives the message. This delay directly induces errors in the estimated $\alpha_{u,v}(k)$ and $\beta_{u,v}(k)$, and thus determines the level of noise in the resulting difference measurement. Therefore, the random delay ultimately affects the time synchronization accuracy. In employing the pairwise synchronization protocol of [3], we subject the message exchanges to a random delay that is Gaussian distributed with mean $150\mu sec$ and standard deviation $10\mu sec$, as these values are considered realistic for wireless sensors networks with current hardware and communication protocols with uncertain-delay elimination (e.g., MAC-layer time-stamping) [8]. Although the relation between the statistics of the random delays and the noise of difference measurements of log-skew and offsets defined in Section II are complex, the noise levels in the difference measurements used are likely to be realistic due to realistic choice of delays.

6.2. Performance in estimating global time

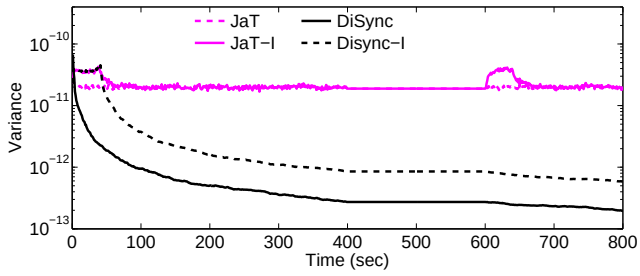
We conduct 1000 Monte Carlo simulations of running algorithms for 800 second (iterations). Figure 3 shows two snapshots of the network during a simulation. As we can see, only a limited number of nodes can communicate with each other. Recall that $a_{uv}(k)=1$ if $(u,v)\in\mathcal{E}(k)$ for all k. The step size function is chosen as $m(x)=\frac{1}{x+3}$. We pick $k_h=k_{\mathcal{H}}=40$ somewhat arbitrarily. Moreover, to evaluate the algorithms under sleep-wake cycle implementation for energy conservation, we force nodes to pause the updates when $k\in[400,600]$. When $k>600$, the step size function is changed to $m(k-k_h-200)$, i.e. it resumes the value before the pause.

Figure 4(a) and 4(b) show the mean and variance of estimation error of the skew of node 3. As expected, both the variances of DiSync and DiSync-I are seen to converge to zero, while that of JaT and JaT-I converge to constant value. In addition, DiSync-I improves the accuracy of the mean of estimation error at the expense of slightly increasing the variance of estimation error.

Figure 5 shows the global time estimation error in one experiment, i.e. $\hat{t}_u(t_k)-t_k$ as a function of t_k for node 3. Both DiSync and DiSync-I show much higher accuracy of global time estimation than that of JaT and JaT-I. The accurate skew estimates is crucial in getting good global time estimates, since even a tiny error in the



(a) Mean



(b) Variance

Fig. 4. Empirically estimated mean and variance of the estimation error of skews in node 3. Note that in (b), y-axis is in logarithm scale.

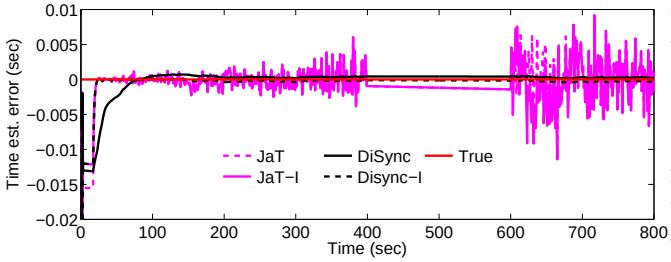


Fig. 5. The estimation error of global time in node 3

skew estimate leads to a large error in the prediction of global time t over time. In addition, the DiSync-I further reduces the initial transient period that DiSync suffer from.

6.3. Comparison with ATS

In ATS [17], each node u estimates the virtual global time using $\hat{t}^r_u(t) = \hat{\varrho}_u(k)\tau_u(t) + \hat{o}_u(k)$ for $t_k \leq t \leq t_{k+1}$, where variables $\hat{\varrho}_u(k)$ and $\hat{o}_u(k)$ can be thought of as the skew and offset of a virtual global time respect to the local time of u during interval k. Here, we present a synchronous version of ATS algorithm that we use in simulation in order to be consistent with the discussion. Each node u updates its $\hat{\varrho}_u(k)$ and $\hat{o}_u(k)$ using $\hat{\varrho}_v(k)$, $\hat{t}^r_v(k)$, $\hat{\alpha}_{uv}(k)$ from its neighbors, where $\hat{\alpha}_{uv}(k)$ is the estimated relative skew. $\hat{\alpha}_{uv}(k)$ is obtained by pairwise communication between u and v during k-th update interval, as part of the ATS algorithm. This is done as follows. Two time-stamped messages are sent from node v to node u: one at the beginning of k-th interval and the other one in the middle of the interval. Note that no return messages from u to v is required. The computation of $\hat{\alpha}_{uv}(k)$ is performed by a low-pass filter as provided in ATS: $\hat{\alpha}_{u,v}(k) = \rho\hat{\alpha}_{u,v}(k-1) + (1-\rho)\frac{\tau_v^{(1)} - \tau_v^{(2)}}{\tau_u^{(1)} - \tau_u^{(2)}}$, where ρ is a tuning parameter and chosen as 0.2 (same value used in ATS). It has been shown that $\lim_{k\to\infty} \alpha_u \hat{\varrho}_u(k) = \bar{\alpha}$, $\lim_{k\to\infty} \hat{o}_u(k) + \beta_u \hat{\varrho}(k) = \bar{\beta}$, where $\bar{\alpha}$ and $\bar{\beta}$ is the skew and offset of the virtual clock with respect to t. The ATS algorithm ensures that the estimated virtual global times in all nodes are eventually equal, i.e., $\lim_{t\to\infty} \hat{t}^r_u(t) = \hat{t}^r_v(t)$ for all u and v.

The performance of ATS is guaranteed under the assumption that the time stamps are exchanged without random delay. To compare with other algorithms under identical conditions, we add random delay to $\tau_u^{(i)}$ for $i = 1, 2$. The delay parameters are the same as those used during the simulation of the other algorithms. In addition, since ATS does not estimate the clock time at any of the nodes, we use the metric "maximum synchronization error" to compare ATS with the other four algorithms. This is defined as $\max_{u,v} |\hat{t}^r_u(t_k) - \hat{t}^r_v(t_k)|$ for ATS, and $\max_{u,v} |\hat{t}_u(t_k) - \hat{t}_v(t_k)|$ for the other four algorithms.

Figure 6 compares maximum synchronization error in one experiment for all five algorithms. Although the maximum synchronization error decreases faster in ATS, JaT and JaT-I at the beginning, the superior robustness to the measurement noise of the DiSync algorithm helps it outperform them after about 100 sec. It can be seen from the figure that the DiSync-I achieves the lowest maximum synchronization error among all algorithms.

VII. Conclusion

We proposed a novel distributed asynchronous algorithm to estimate clock skews and offsets from pairwise difference measurements of log-skews and offsets. The nodes measure log-skew difference and offset difference with nearby neighbors by exchanging time stamped messages. A node fuses these measurements with current estimates to iteratively

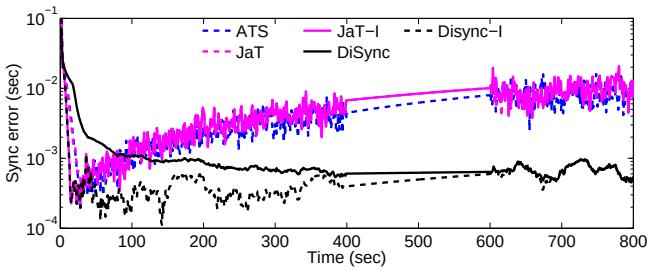


Fig. 6. Maximum synchronization error along time in one experiment.

update their estimates of skew and offset. The algorithm is inspired by the recent work on using stochastic approximation in consensus algorithms. The variance of skew/offset estimation error asymptotically converges to zero under certain assumptions. Using estimated skews and offsets, nodes can estimate the time of global clock accurately which was demonstrated in numerical evaluation. Simulations also show that the proposed algorithms outperform competing algorithms.

The fact that clocks are not synchronized poses an unique challenge in applying stochastic approximation ideas to log-skew and offset estimation problem, since all nodes need to reduce a gain in a synchronous manner. This was addressed by providing an iteration schedule to the nodes ahead of time so that all nodes can effectively perform updates synchronously. The scheduled iteration interval grows along time, though the growth rate is quite slow. In many cases the growing interval between iterations schedules may be useful since it automatically slows down the rate of synchronization over time. This has the desired effect of more iterations and faster lowering of synchronization error initially at the cost of high communication overhead, while lowering the overhead as more accurate synchronization is achieved. In cases when the growth of synchronization interval is undesirable, the reference nodes could restart the synchronization process. Another possibility is to use the estimated skews after some time has passed instead of using pre-specified bounds at all times. Since the proposed algorithms provide highly accurate skew estimates, doing so will ensure synchronous updates while keeping the time interval between updates from growing. The effectiveness of this strategy will be studied in future work. There are many additional avenues of further exploration, such as the role of network topology [21, 26] and design parameters $k_{\mathcal{H}}$,k_h on the convergence rate.

REFERENCES

[1] B. M. Sadler and A. Swami, "synchronization in sensor networks: an overview," in *IEEE MILCOM*, October 2006, pp. 1–6.

[2] J. R. Vig, "Introduction to quartz frequency standards," Army Research Laboratory, Tech. Rep., 1992.

[3] K.-L. Noh, Q. M. Chaudhari, E. Serpedin, and B. W. Suter, "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 766–777, Apr 2007.

[4] S. Yoon, C. Veerarittiphan, and M. L. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 2, pp. 1–34, Jun 2007.

[5] M. Leng and Y.-C. Wu, "On clock synchronization algorithms for wireless sensor networks under unknown delay," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 182–190, Jan 2010.

[6] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[8] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[9] P. Barooah and J. P. Hespanha, "Estimation from relative measurements: Error bounds from electrical analogy," in *Proc. of the 2nd International Conference on Intelligent Sensing and Information Processing(ICISIP)*, January 2005, pp. 88–93.

[10] P. Barooah, N. M. da Silva, and J. P. Hespanha, "Distributed optimal estimation from relative measurements for localization and time synchronization," in *International Conference on Distributed Computing in Sensor Systems DCOSS'06*, San Francisco, June 2006, pp. 266 – 281.

[11] A. Giridhar and P. R. Kumar, "Distributed clock synchronization in wireless networks: Algorithms and analysis (I)," in *45th IEEE Conference on Decison and Control*, December 2006, pp. 4915 – 4920.

[12] R. Solis, V. S. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *Proc. of the 45th IEEE Conference on Decison and Control*, December 2006, pp. 2734–2739.

[13] C. Liao and P. Barooah, "Distributed clock skew and offset estimation from relative measurements in mobile networks with markovian switching topology," *Automatica*, vol. 49, no. 10, pp. 3015 – 3022, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109813003634

[14] M. Huang, S. Dey, G. N. Nair, and J. H. Manton, "Stochastic consensus over noisy networks with Markovian and arbitrary switches," *Automatica*, vol. 46, no. 10, pp. 1571–1583, Oct. 2010.

[15] T. Li and J. Zhang, "Consensus conditions of multi-agent systems with time-varying topologies and stochastic communication noises," *Automatic Control, IEEE Transactions on*, vol. 55, no. 9, pp. 2043–2057, 2010.

[16] C. Liao and P. Barooah, "Time synchronization in mobile sensor networks from difference measurements," in *In proceedings of the 49th IEEE Conference on Decision and Control*, December 2010, pp. 2118 – 2123.

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

[18] R. Carli, E. D'Elia, and S. Zampieri, "A pi controller based on asymmetric gossip communications for clocks synchronization in wireless sensors networks," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, dec. 2011, pp. 7512 –7517.

[19] C. Liao and P. Barooah, "Di-Sync: High-accuracy distributed clock synchronization in mobile ad-hoc networks from noisy relative measurements," in *American Control Conference*, June 2013, pp. 3338–3343.

[20] ——, "Estimation from relative measurements in mobile networks with markovian switching topology: Clock skew and offset estimation for time synchronization," *ArXiV preprint*, 2013. [Online]. Available: http://arxiv.org/abs/1301.2218

[21] P. Barooah and J. P. Hespanha, "Graph effective resistances and distributed control: Spectral properties and applications," in *Proc. of the 45th IEEE Conference on Decision and Control*, December 2006, pp. 3479–3485.

[22] C. Liao, "Distributed time synchronization from relative measurement in mobile wireless sensor networks," Ph.D. dissertation, University of Florida, April 2013.

[23] L. Gyrfi, "Stochastic approximation from ergodic sample for linear regression," *Probability Theory and Related Fields*, vol. 54, pp. 47–55, 1980.

[24] M. Kouritzin, "On the convergence of linear stochastic approximation procedures," *Information Theory, IEEE Transactions on*, vol. 42, no. 4, pp. 1305 –1309, jul 1996.

[25] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, Aug 2002.

[26] M. Pirani and S. Sundaram, "On the smallest eigenvalue of grounded laplacian matrices," *IEEE Transactions on Automatic Control*, 2015.