Automatica 92 (2018) 133–142

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

A fast clock synchronization algorithm for wireless sensor networks[☆]

Kan Xie [a], Qianqian Cai [a,*], Minyue Fu [b,a]

[a] School of Automation, Guangdong University of Technology, and Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China
[b] School of Electrical Engineering and Computer Science, The University of Newcastle, NSW 2308, Australia

A R T I C L E   I N F O

Article history:
Received 21 May 2016
Received in revised form 18 September 2017
Accepted 13 December 2017
Available online 23 March 2018

Keywords:
Wireless sensor networks
Clock synchronization
Average consensus
Consensus control
Distributed control

A B S T R A C T

This paper proposes a novel clock synchronization algorithm for wireless sensor networks (WSNs). The algorithm is derived using a fast finite-time average consensus idea, and is *fully distributed*, meaning that each node relies only on its local clock readings and reading announcements from its neighbours. For networks with an acyclic graph, the algorithm converges in only d iterations for clock rate synchronization and another d iterations for clock offset synchronization, where d is the graph diameter. The algorithm enjoys low computational and communicational complexities and robustness against transmission adversaries. Each node can execute the algorithm asynchronously without the need for global coordination. Due to its fast convergence, the algorithm is most suitable for large-scale WSNs. For WSNs with a cyclic graph, a fast distributed depth-first-search (DFS) algorithm can be applied first to form a spanning tree before applying the proposed synchronization algorithm.

1. Introduction

Rapid technological advances on wireless sensor design and manufacturing have enabled wide applications of wireless sensor networks (WSNs) in various fields including surveillance, environmental monitoring, traffic monitoring, industrial automation, autonomous vehicles, smart grid, transportation networks, and so on. Wireless sensors are typically equipped with low-quality crystals (due to low cost) and have stringent energy constraints, and they are usually deployed in an ad hoc fashion. One of the great challenges for WSNs is how to synchronize the sensor clocks. This problem becomes more paramount as the size of the network gets larger.

Unlike wired networks, such as the internet, which can use the *network Time Protocol* (NTP) (Mills, 1991) to synchronize clocks in a hierarchical way by using primary and secondary time servers, WSNs cannot adopt this kind of synchronization approach due to energy consumption and bandwidth constraints (Sundararaman, Buy, & Kshemkalyani, 2005). Solutions which rely on accurate reference clocks or expensive signalling sources (such as GPS signalling) are inappropriate due to cost and energy constraints as well. Many conventional clock synchronization schemes are not suitable for WSNs; see El Khediri, Nasri, Samet, Wei, and Kachouri (2012), Rhee, Lee, Kim, Serpedin, and Wu (2009), Sarvghadi and Wan (2014) and Sundararaman et al. (2005) for overviews on clock synchronization for WSNs.

Three clock synchronization frameworks are available: *master–slave, peer-to-peer, and distributed*. Synchronization is usually done by either aligning the clock readings (called *clock offset synchronization*, or simply *clock synchronization* in many references) or aligning the clock rates (called *clock rate synchronization*, or *skew compensation*) or both. The so-called *drift compensation* (aligning the rate of a clock rate) is rarely done.

In a master–slave synchronization scheme, a "master" node is chosen as the global reference clock and all other nodes are treated as "slaves". Protocols for clock offset synchronization include *flooding* schemes (Ferrari, Zimmerling, Thiele, & Saukh, 2011), IEEE 802.11 based clock synchronization protocol (Mock, Frings, Nett, & Trikaliotis, 2000), *Delay Measurement Time Synchronization* (DMTS) (Ping, 2003), and *Pairwise Broadcast Synchronization* (PBS) (Noh, Serpedin, & Qaraqe, 2008). Clock rate synchronization can also be done under the master–slave framework. Protocols of this kind include *Flooding Time Synchronization Protocol* (FTSP) (Maróti, Kusy, Simon, & Lédeczi, 2004), Tiny-Sync (Yoon, Veerarittiphan, & Sichitiu, 2007), a maximum likelihood estimator-based scheme (Chaudhari, Serpedin, & Qaraqe, 2008), and feedback control based approach (e.g., PI control (Chen, Yu, Zhang, Chen, & Sun, 2010), FLOPSYNC (Leva, Terraneo, Rinaldi, Papadopoulos, & Maggio, 2016), *Self-Correcting Time Synchronization* (SCTS) protocol (Ren, Lin, & Liu, 2008), and an asymmetric gossip communication algorithm (Carli, D'Elia, & Zampieri, 2011)).

[☆] This work was supported by the National Natural Science Foundation of China (Grant No. 61633014 and U1701264). The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Claudio De Persis under the direction of Editor Christos G. Cassandras.

[*] Corresponding author.
E-mail addresses: kanxiegdut@gmail.com (K. Xie), qianqian.cai@outlook.com (Q. Cai), minyue.fu@newcastle.edu.au (M. Fu).

https://doi.org/10.1016/j.automatica.2018.03.004
0005-1098/© 2018 Elsevier Ltd. All rights reserved.

Peer-to-peer synchronization schemes assume that any node can communicate to any other node directly. Protocols of this kind include *Reference Broadcast Synchronization* (RBS) (Elson, Girod, & Estrin, 2002), *Tiny-Sync* and *Mini-Sync* (TS/MS) (Sichitiu & Simple, 2003), *Timing-Sync Protocol for Sensor Networks* (TPSN) (Ganeriwal, Kumar, & Srivastava, 2003), *Lightweight Time Synchronization* (LTS) (Van Greunen & Rabaey, 2003), *TSync* (Dai & Han, 2004). Although these schemes eliminate the risk of master node failure, they are suitable for small networks only (Sundararaman et al., 2005).

In a distributed synchronization scheme, no master or global clock is assumed and each node is allowed to communicate with only neighbouring nodes. The distributed approach has been widely studied for many estimation and control applications (Cortés, 2006; Hendrickx et al., 2004; Kashyap, Basar, & Srikant, 2007; Li, Fu, Xie, & Zhang, 2011; Xiao & Boyd, 2004; Xie, Cai, Zhang, & Fu, 2018). Apart from the major attraction of having no single point of failure and each node being autonomous, the distributed approach tends to enjoy nice properties including algorithmic simplicity, resilience against network adversaries, topological changes, and scalability to large networks. Unfortunately, not many distributed algorithms for clock synchronizations are available so far. A prominent approach is the average consensus-based design, including the *Average TimeSync* (AST) protocol (Carli, Chiuso, Schenato, & Zampieri, 2011; Carli & Zampieri, 2014; Kadowaki & Ishii, 2015; Schenato & Fiorentin, 2011; Simeone & Spagnolini, 2007). These algorithms enjoy simplicity, but the main drawback is that consensus is achieved only asymptotically, requiring too many iterations of computations and communications in practice. The *belief propagation* (or *message passing*) algorithms (Ahmad, Zennaro, Serpedin, & Vangelista, 2012; Du & Wu, 2013; Leng & Wu, 2011; Zennaro et al., 2013) have good accuracies and can be implemented asynchronously. A recursive least-squares estimation scheme is used in Solis, Borkar, and Kumar (2006) for multi-hop WSNs. In Bolognani, Carli, Lovisari, and Zampieri (2016), a randomized distributed algorithm is given to achieve clock synchronization. A distributed Kalman filter is used in Luo and Wu (2013) to track clock parameters.

In this paper, we study the problem of distributed clock synchronization for large WSNs. Our general approach is similar to those in average consensus-based algorithms (Carli et al., 2011; Carli & Zampieri, 2014; Kadowaki & Ishii, 2015; Schenato & Fiorentin, 2011; Simeone & Spagnolini, 2007), but with the main aim of coming up with a fast algorithm for consensus. To allow solutions *truly scalable* to large WSNs, we need to ensure: (1) In each iteration of the algorithm, the available information at each node should be limited to its own measurements and information exchanged with its direct neighbours; (2) No master clock is assumed and no synchronous sampling is used for all the nodes (which would otherwise imply the existence of a master clock); (3) The complexities of the algorithm should be bounded for each node and each iteration, not growing as the size of the network grows. Our clock synchronization properties include:

- *Rate synchronization*: All the local clocks should be synchronized to a virtual clock with the rate equal to the *geometric mean* of the all the local clock rates;
- *Offset synchronization*: After synchronization, all the local clocks should achieve the same clock offset (i.e., the same clock reading at any global time instant);
- *Continuous transition*: Each compensated local clock reading should be continuous in time;
- *Minimum clock rate*: Each compensated local clock should have a guaranteed minimum rate during the transition (to avoid clock stalling or time reversal);
- *Finite-time Convergence*: Synchronization (for both rate and offset) should be completed in a prescribed time period.

We propose a new fast clock synchronization algorithm for both clock rate synchronization and clock offset synchronization. The algorithm is *fully distributed* with the above synchronization properties. We first consider WSNs with an acyclic graph (i.e., tree graph) and give our algorithm in two parts: clock rate synchronization (Algorithm 1) and clock offset synchronization (Algorithm 2). Each part takes only d iterations, where d is the graph diameter. The algorithm is resilient to transmission delay and packet loss and admits asynchronous implementation. We then consider general sensor networks (cyclic or not) and apply a fast distributed depth-first-search (DFS) algorithm (Xie, Cai, Zhang, & Fu, 2018) (Algorithm 0) first to construct and verify a spanning tree. Combined with Algorithm 0, proposed Algorithms 1 and 2 apply to any WSN with a connected and undirected graph, due to the fact that such a graph always has a spanning tree.

The main advantage of the proposed algorithms is that we can achieve fast and finite-time convergence for clock synchronization, where the Laplacian matrix based approach gives only asymptotic convergence. Our convergence time depends on the graph diameter and is independent of the graph topology, unlike the Laplacian matrix based approach which is greatly influenced by the eigenvalues of the Laplacian matrix (Li et al., 2011).

The rest of the paper is organized as follows: Section 2 formulates the clock synchronization problem; Section 3 gives the proposed algorithm; Section 4 discusses the properties and modifications; Section 5 gives three examples; Section 6 concludes the paper.

2. Problem formulation

Consider a WSN with n sensing nodes. We model the WSN using an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with a set of nodes $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of edges $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}\}$. A graph is called *undirected* if information can flow in two ways between the two nodes on any edge. A graph is called *acyclic* if it is *connected* and has no loops, i.e., it is a *tree graph*. Denote by $\mathcal{N}_i$ the set of neighbouring nodes connected to node i, and denote by $|\mathcal{N}_i|$ the cardinality of $\mathcal{N}_i$. We focus on large graphs with sparse connectivity and the property that $|\mathcal{N}_i| \ll n$. The local clock model for each sensor (node) $i \in \mathcal{V}$ is given by

$$x_i(t) = a_i t + b_i \tag{1}$$

where t is the global time, $a_i > 0$ represents the clock rate, b_i is the initial time (i.e., value of $x_i(0)$) and t represents the global time. We denote by one *unit of local time* the time takes for $x_i(t)$ to advance from one integer to the next (adjacent) integer. That is, one unit of global time equals a_i units of local time for node i. Without loss of generality, we will call one unit of time one *minute*.

We emphasize that the parameters a_i and b_i and the global time t are all *unknown* to all the nodes. However, each local clock announces (i.e., broadcasts) its local time $\tau \in \mathcal{Z}$ (the set of integers) every local minute. It is assumed that only the neighbouring nodes will receive these announcements. It is also assumed that transmission time between nodes is negligible. This assumption is reasonable for WSNs and is commonly used; see, e.g., Carli and Zampieri (2014) and Schenato and Fiorentin (2011). In addition, transmission delays can be compensated, using, e.g., the conventional Network Time Protocol (NTP) (Mills, 1991) or a standard flooding protocol (Ferrari et al., 2011) with minor communication overhead between neighbouring nodes in the clock measurement process.

Denote by $t_j(\tau)$ the global time instant at which node j announces its τ-th *local minute*, i.e., $x_j(t_j(\tau)) = \tau$. Due to the assumption that the transmission time between nodes is negligible, node $i \in \mathcal{N}_j$ will receive this announcement τ at its own local time

$x_i(t_j(\tau))$. We assume in the sequel that, for each $\tau \in \mathcal{Z}$, only the readings of $x_i(t_j(\tau)), j \in \mathcal{N}_i$, are available to node i.

The problem of *distributed clock synchronization* is stated as follows: Design a *compensator* (function) $f_i(\cdot)$ for each node $i \in \mathcal{V}$ using its local clock readings and the announcements from its neighbouring set $\mathcal{N}_i$ such that the compensated local clock $\hat{x}_i(t) = f_i(x_i(t))$ is synchronized in the sense that

$$\hat{x}_i(t) = at + b, \quad \forall t \geq T \tag{2}$$

for some common values of $a > 0$ and b after a pre-specified period of time T.

The clock synchronization problem is further split into *rate synchronization* problem (also known as frequency synchronization) for which only a common rate needs to be achieved, and *offset synchronization* problem (also known as phase synchronization) for which a common initial time b needs to be achieved, assuming rate synchronization is achieved. We impose a further "fairness" requirement that a be the geometric mean of a_i (i.e., $a = \sqrt[n]{a_1 a_2 \ldots a_n}$).

To address the complexity property for scalability as discussed in Introduction, we impose the following *constraints* on the algorithm's complexities:

(1) Information exchange: Each node i can communicate with each $j \in \mathcal{N}_i$ only once per iteration.
(2) Computation: Each node i's computational load should be at most $O(|\mathcal{N}_i|)$ per iteration.
(3) Storage: Each node i's storage should be at most $O(|\mathcal{N}_i|)$ over all iterations.

The above constraints ensure that the complexities of the algorithm per node per iteration will not increase as the size of the network increases. To see the relationship between the network size and the required number of iterations, we give the following *benchmark* result.

Lemma 1. *Under the local information exchange constraint above, a connected undirected graph $\mathcal{G}$ with diameter d needs at least d iterations to achieve either clock rate synchronization or clock offset synchronization.*

Proof. Let nodes i and j be such that they are d hops away from each other (such nodes exist by the definition of d). The information at node i needs to propagate to node j in order for node j to correctly align the clock rates or clock offsets for all the sensors. By the local information exchange constraint, this will take at least d iterations.

Note that the minimum iteration number given above is independent of the algorithm and the network topology. We will show in the next section that our proposed algorithm achieves this minimum number.

3. Distributed algorithm for clock synchronization

In this section, we introduce our distributed algorithm for clock synchronization, provide its key property on acyclic graphs and analyse its complexities. The proposed synchronization algorithm has two parts: one for clock rate and one for clock offset.

3.1. Clock rate synchronization

Node i can compute the local time difference $x_i(t_j(\tau)) - x_i(t_j(\tau - 1))$ between two adjacent announcements from node j. Following Carli and Zampieri (2014) and Schenato and Fiorentin (2011), we define

$$\delta_{ij}(\tau) = \ln(x_i(t_j(\tau)) - x_i(t_j(\tau - 1))), \tag{3}$$

and note that $\{\delta_{ij}(\tau) : j \in \mathcal{N}_i\}$ is a set available to node i. We have the following result.

Lemma 2. *Given any $i \in \mathcal{V}, j \in \mathcal{N}_i$, it holds that*

$$\delta_{ij}(\tau) = \alpha_i - \alpha_j \tag{4}$$

for any $\tau \in \mathcal{Z}$, where $\alpha_i = \ln a_i$ and $\alpha_j = \ln a_j$.

Proof. By definition, $x_j(t_j(\tau)) = \tau$. Because the transmission time between neighbouring nodes is negligible, node $i \in \mathcal{N}_j$ receives this announcement immediately and can check its own local time $x_i(t_j(\tau))$. From (1),

$$x_i(t_j(\tau)) - x_i(t_j(\tau - 1)) = a_i(t_j(\tau) - t_j(\tau - 1));$$
$$x_j(t_j(\tau)) - x_j(t_j(\tau - 1)) = a_j(t_j(\tau) - t_j(\tau - 1)).$$

Since $x_j(t_j(\tau)) - x_j(t_j(\tau - 1)) = \tau - (\tau - 1) = 1$, it follows from the above that

$$x_i(t_j(\tau)) - x_i(t_j(\tau - 1)) = a_i/a_j > 0.$$

Then, (4) follows from $\delta_{ij}(\tau) = \ln(x_i(t_j(\tau)) - x_i(t_j(\tau - 1)))$ and $\alpha_i = \ln a_i$.

Algorithm 1 is our proposed distributed algorithm for clock rate synchronization, which is executed once for each time τ. The available information at each node i is $\delta_{ij}(\tau)$. For notational brevity, the dependence on τ is suppressed. The algorithm iterates d times and aims to compute two variables at each iteration k: $s_i(k)$ represents the number of nodes in $\mathcal{G}$ involved in clock synchronization for node i, and $\eta_i(k)$ represents the compensated rate deviation for node i.

Algorithm 1 (Clock Rate Synchronization)

- **Initialization:** At each node i: For each $j \in \mathcal{N}_i$, set $\eta_{i\to j}(0) = 0$, $s_{i\to j}(0) = 1$ and transmit them to node j; Then, store the received $\eta_{j\to i}(0)$ and $s_{j\to i}(0)$ from each $j \in \mathcal{N}_i$.

- **Main loop:** At *iteration* $k = 1, 2, \cdots, d$, for each node i, compute

$$s_i(k) = 1 + \sum_{j\in\mathcal{N}_i} s_{j\to i}(k-1) \tag{5}$$

$$\tilde{\eta}_i(k) = \sum_{j\in\mathcal{N}_i} (s_{j\to i}(k-1)\delta_{ij} + \eta_{j\to i}(k-1)) \tag{6}$$

then for each $j \in \mathcal{N}_i$, compute

$$s_{i\to j}(k) = s_i(k) - s_{j\to i}(k-1) \tag{7}$$
$$\eta_{i\to j}(k) = \tilde{\eta}_i(k) - (s_{j\to i}(k-1)\delta_{ij} + \eta_{j\to i}(k-1)). \tag{8}$$

If $k < d$, transmit the above two quantities to node j. Then, store the received $\eta_{j\to i}(k)$ and $s_{j\to i}(k)$ from each $j \in \mathcal{N}_i$. If $k = d$, compute

$$\eta_i(k) = -\tilde{\eta}_i(k)/s_i(k). \tag{9}$$

Theorem 3. *Suppose the graph $\mathcal{G}$ is undirected and acyclic with diameter d. Then, we have*

$$s_i(k) = 1 + |\mathcal{V}_i(k)| \tag{10}$$

$$\tilde{\eta}_i(k) = |\mathcal{V}_i(k)|\alpha_i - \sum_{j\in\mathcal{V}_i(k)} \alpha_j \tag{11}$$

for $k = 1, 2, \ldots, d$, where $\mathcal{V}_i(k)$ is the set of nodes in $\mathcal{G}$ that are at most k hops away from node i (excluding node i), and $|\mathcal{V}_i(k)|$ denotes its cardinality. Consequently, $s_i(k)$ and $\tilde{\eta}_i(k)$ converge after d iterations and we have

$$\eta_i(k) = \alpha_i - \bar{\alpha}, \quad \forall\, k \geq d, \; i \in \mathcal{V}, \tag{12}$$

where $\bar{\alpha}$ is the arithmetic mean of α_i, $i \in \mathcal{V}$. The rate-synchronized local clock for each node $i \in \mathcal{V}$ is given by

$$\tilde{x}_i(t) = \exp(-\eta_i(d))(x_i(t) - \tau) + \tau \tag{13}$$

which has the property of

$$\tilde{x}_i(t) = at + \beta_i, \tag{14}$$

where $a = \sqrt[n]{a_1 a_2 \dots a_n}$ and

$$\beta_i = -\frac{a}{a_i}(b_i - \tau) + \tau. \tag{15}$$

Proof. For each edge $(i, j) \in \mathcal{E}$, we consider the convergence of $s_{i\to j}(k)$ and $\eta_{i\to j}(k)$. To do so, we construct the disjoint subgraphs $\mathcal{G}_i$ (containing node i) and $\mathcal{G}_j$ (containing node j) of $\mathcal{G}$ by removing the edge (i, j). Note that $\mathcal{G}$ is the union of $\mathcal{G}_i$ and $\mathcal{G}_j$ plus the edge (i, j), due to its acyclic nature. Denote by $\mathcal{V}_i$ (reps. $\mathcal{V}_j$) the set of nodes in $\mathcal{G}_i$ (reps. $\mathcal{G}_j$). We see from (7)–(8) that $s_{i\to j}(k)$ and $\eta_{i\to j}(k)$ are constructed using the information in $\mathcal{G}_i$ only because the information flow from node j to node i gets removed in each iteration. Recall that at the Initialization step ($k = 0$), we set $s_{i\to j}(0) = 1$ and $\eta_{i\to j}(0) = 0$. Next, consider the case of $k = 1$. From (5) and (7), $s_{i\to j}(1)$ contains all $s_{m\to i}(0)$ for all the neighbouring nodes m, except node j. Using the definition of $\mathcal{V}_i(k)$, we get

$$s_{i\to j}(1) = 1 + \sum_{m\in\mathcal{V}_i(1)\setminus\mathcal{V}_j} 1 = 1 + |\mathcal{V}_i(1) \setminus \mathcal{V}_j|.$$

That is, $(s_{i\to j}(1) - 1)$ equals the number of all the nodes 1 hop away from node i, except nodes in $\mathcal{V}_j$. Similarly, noting that $\mathcal{N}_i = \mathcal{V}_i(1)$, (6) and (8) give

$$\begin{aligned}\eta_{i\to j}(1) &= \sum_{m\in\mathcal{N}_i\setminus\{j\}} (s_{m\to i}(0)\delta_{im} + \eta_{m\to i}(0))\\ &= |\mathcal{V}_i(1) \setminus \mathcal{V}_j|\alpha_i - \sum_{m\in\mathcal{V}_i(1)\setminus\mathcal{V}_j} \alpha_m.\end{aligned}$$

For $k = 2$, we have

$$\begin{aligned}s_{i\to j}(2) &= 1 + \sum_{m\in\mathcal{N}_i\setminus\mathcal{V}_j} s_{m\to i}(1)\\ &= 1 + \sum_{m\in\mathcal{N}_i\setminus\mathcal{V}_j} 1 + \sum_{m\in\mathcal{N}_i\setminus\mathcal{V}_j} |\mathcal{V}_m(1) \setminus \mathcal{V}_i|.\end{aligned}$$

The first sum above is equal to $|\mathcal{V}_i(1)\setminus\mathcal{V}_j|$. In the minute sum above, each $|\mathcal{V}_m(1) \setminus \mathcal{V}_i|$ term is equal to the number of nodes 1 hop away from node m, except those in $\mathcal{V}_i$, which means that this term is equal to the number of nodes 2 hops away from node i, except those in $\mathcal{V}_j$. Hence, we get

$$s_{i\to j}(2) = 1 + |\mathcal{V}_i(2) \setminus \mathcal{V}_j|,$$

hence $(s_{i\to j}(2) - 1)$ equals the number of nodes at most 2 hops away from node i, except those in $\mathcal{V}_j$. Likewise,

$$\begin{aligned}\eta_{i\to j}(2) &= \sum_{m\in\mathcal{N}_i\setminus\{j\}} (s_{m\to i}(1)\delta_{im} + \eta_{m\to i}(1))\\ &= \sum_{m\in\mathcal{N}_i\setminus\{j\}} (1 + |\mathcal{V}_m(1) \setminus \mathcal{V}_i|)(\alpha_i - \alpha_m)\\ &\quad + \sum_{m\in\mathcal{N}_i\setminus\{j\}} (|\mathcal{V}_m(1) \setminus \mathcal{V}_i|\alpha_m - \sum_{u\in\mathcal{V}_m(1)\setminus\mathcal{V}_i} \alpha_u)\\ &= |\mathcal{V}_i(2) \setminus \mathcal{V}_j|\alpha_i - \sum_{m\in\mathcal{N}_i\setminus\{j\}} \left(\alpha_m - \sum_{u\in\mathcal{V}_m(1)\setminus\mathcal{V}_i} \alpha_u\right)\\ &= |\mathcal{V}_i(2) \setminus \mathcal{V}_j|\alpha_i - \sum_{m\in\mathcal{V}_i(2)\setminus\mathcal{V}_j} \alpha_m.\end{aligned}$$

Repeating the above, we get, for a general k,

$$\begin{aligned}s_{i\to j}(k) &= 1 + |\mathcal{V}_i(k) \setminus \mathcal{V}_j|\\ \eta_{i\to j}(k) &= |\mathcal{V}_i(k) \setminus \mathcal{V}_j|\alpha_i - \sum_{m\in\mathcal{V}_i(k)\setminus\mathcal{V}_j} \alpha_m.\end{aligned}$$

Next, we consider $s_i(k)$ and $\tilde{\eta}_i(k)$. Note from (7) that $s_i(k) = s_{i\to j}(k) + s_{j\to i}(k-1)$. Also note that node j is 1 hop away from node i, which means that all the nodes in $\mathcal{G}_j$ that are $k-1$ hops away from node j are actually k hops away from node i, when considering the graph $\mathcal{G}$:

$$(\mathcal{V}_i(k) \setminus \mathcal{V}_j) \cup (\mathcal{V}_j(k-1) \setminus \mathcal{V}_i) \cup \{j\} = \mathcal{V}_i(k).$$

It follows that

$$\begin{aligned}s_i(k) &= (1 + |\mathcal{V}_i(k) \setminus \mathcal{V}_j|) + (1 + |\mathcal{V}_j(k-1) \setminus \mathcal{V}_i|)\\ &= 1 + |\mathcal{V}_i(k)|,\end{aligned}$$

which is (10). Eq. (11) is shown in the same way. It then follows that all the updating stops when $k \geq d$, and for such k, we get

$$\begin{aligned}s_i(k) &= s_i(d) = 1 + |\mathcal{V}_i(d)| = 1 + |\mathcal{V} \setminus \{i\}| = |\mathcal{V}|\\ \tilde{\eta}_i(k) &= \tilde{\eta}_i(d) = |\mathcal{V} \setminus \{i\}|\alpha_i - \sum_{j\in\mathcal{V}\setminus\{i\}} \alpha_j = |\mathcal{V}|\alpha_i - \sum_{j\in\mathcal{V}} \alpha_j,\end{aligned}$$

resulting in (12) for each node i.

It remains to check that after iteration d, all the updating will stop. To see this, consider any path p in $\mathcal{G}$ going through nodes i and j. This path is split into three segments: path p_i in $\mathcal{G}_i$, edge (i, j) and path p_j in $\mathcal{G}_j$. Denote the maximum path length of p_i by d_i and the maximum path length of p_j by d_j. From the analysis above, we see that $s_{i\to j}(k)$ and $\eta_{i\to j}(k)$ will converge after d_i iterations, and that $s_{j\to i}(k)$ and $\eta_{j\to i}(k)$ will converge after d_j iterations. Since the maximum path length of p is d, we have $d_i + d_j + 1 \leq d$. In particular, $d \geq d_i$ and $d - 1 \geq d_j$. Hence, at iteration d, all the terms $s_{i\to j}(d)$, $\eta_{i\to j}(d)$, $s_{j\to i}(d-1)$, $\eta_{j\to i}(d-1)$ must have converged. From (7)–(8), the above means that $s_i(d)$ and $\tilde{\eta}_i(d)$ must have converged.

Finally, substituting $\exp(-\eta_i(d)) = a/a_i$ and (1) into (13) yields (14) immediately.

Remark 4. We see from Theorem 3 that the converged common clock rate is the *geometric mean* of all the local clock rates, i.e., $a = \sqrt[n]{a_1 a_2 \dots a_n}$. Because of this, we can say that Algorithm 1 achieves average consensus in the geometric sense. This is different from many algorithms in the literature which aim at reaching the arithmetic mean $(a_1 + a_2 + \cdots + a_n)/n$. The difference is negligible in practice because a typical sensor node has a drift of up to 8 (i.e., ± 4) seconds only every 24 h (see more details in Section 5.3 later). If re-synchronization is done every 2 min, the drift is up to ± 0.0056 second.

3.2. Clock offset synchronization

Given any node i and its neighbouring node $j \in \mathcal{N}_i$, define the offset difference $\beta_{ij} = \beta_i - \beta_j$. We show below that β_{ij} can be computed locally.

Lemma 5. *For every node $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$, β_{ij} is available at node i and it is given by*

$$\beta_{ij} = \exp(-\eta_i(d))(x_i(t_j(\tau)) - \tau). \tag{16}$$

Proof. Since $x_j(t_j(\tau)) = \tau$, we have

$$\tilde{x}_j(t_j(\tau)) = \exp(-\eta_j(d))(x_j(t_j(\tau_j)) - \tau) + \tau = \tau.$$

It follows that

$$\begin{aligned}\beta_{ij} &= \beta_i - \beta_j = \tilde{x}_i(t_j(\tau)) - \tilde{x}_j(t_j(\tau))\\ &= \{\exp(-\eta_i(d))(x_i(t_j(\tau)) - \tau) + \tau\} - \tau\\ &= \exp(-\eta_i(d))(x_i(t_j(\tau)) - \tau).\end{aligned}$$

Note that the information of β_{ij} is available at node i because $x_i(t_j(\tau))$ is the local time at node i at which node j announces its local time of τ minutes.

Algorithm 2 is our proposed distributed algorithm for clock offset synchronization, also executed at each time τ, after Algorithm 1. It also iterates d times. For each iteration k, it aims to compute $s_i(k)$ as before, and $\gamma_i(k)$ which is the compensated offset deviation for node i.

Algorithm 2 (Clock Offset Synchronization)

- **Initialization:** At each node i: For each $j \in \mathcal{N}_i$, set $\gamma_{i\to j}(0) = 0$, $s_{i\to j}(0) = 1$ and transmit them to node j; Then, store the received $\gamma_{j\to i}(0)$ and $s_{j\to i}(0)$ from each $j \in \mathcal{N}_i$.
- **Main loop:** At iteration $k = 1, 2, \cdots, d$, for each node i, compute

$$s_i(k) = 1 + \sum_{j\in\mathcal{N}_i} s_{j\to i}(k-1) \tag{17}$$

$$\tilde{\gamma}_i(k) = \sum_{j\in\mathcal{N}_i} (s_{j\to i}(k-1)\beta_{ij} + \gamma_{j\to i}(k-1)) \tag{18}$$

then for each $j \in \mathcal{N}_i$, compute

$$s_{i\to j}(k) = s_i(k) - s_{j\to i}(k-1) \tag{19}$$

$$\gamma_{i\to j}(k) = \tilde{\gamma}_i(k) - (s_{j\to i}(k-1)\beta_{ij} + \gamma_{j\to i}(k-1)). \tag{20}$$

If $k < d$, transmit the above two quantities to node j. Then, store the received $\gamma_{j\to i}(k)$ and $s_{j\to i}(k)$ from each $j \in \mathcal{N}_i$. If $k = d$, compute

$$\gamma_i(k) = -\tilde{\gamma}_i(k)/s_i(k). \tag{21}$$

Theorem 6. *Suppose the graph $\mathcal{G}$ is undirected and acyclic with diameter d and Algorithm 2 is applied. Then, we have (10) and*

$$\tilde{\gamma}_i(k) = |\mathcal{V}_i(k)|\beta_i - \sum_{j\in\mathcal{V}_i(k)} \beta_j \tag{22}$$

for $k = 1, 2, \ldots, d$, where $\mathcal{V}_i(k)$ are defined in Theorem 3. Consequently, $s_i(k)$ and $\tilde{\gamma}_i(k)$ converge after d iterations, and we have

$$\gamma_i(k) = \beta_i - b, \quad \forall\, k \geq d,\ i \in \mathcal{V}, \tag{23}$$

where b is the arithmetic mean of β_i, $i \in \mathcal{V}$. The synchronized local clock for node i is given by

$$\check{x}_i(t) = \exp(-\eta_i(d))(x_i(t) - \tau) + \tau - \gamma_i(d) \tag{24}$$

for all $t \geq t_i(\tau)$, which has the property of

$$\check{x}_i(t) = at + b,\ \forall t \geq t_i(\tau) \tag{25}$$

where $a = \sqrt[n]{a_1 a_2 \ldots a_n}$ and $b = (\beta_1 + \beta_2 + \cdots \beta_n)/n$.

Proof. The proof is identical to that of Theorem 3 when we replace $(\delta_{ij}, \tilde{\eta}_i, \eta_i, \alpha_i, \bar{\alpha})$ with $(\beta_{ij}, \tilde{\gamma}_i, \gamma_i, \beta_i, b)$. Using $\check{x}_i(t) = \tilde{x}_i(t) - \gamma_i(d)$ and substituting (14) and (23) into (24) gives (25).

Remark 7. We see from Theorem 6 that the converged common offset b is the arithmetic mean of the offsets β_i, i.e., $b = (\beta_1 + \beta_2 + \cdots + \beta_n)/n$. That is, Algorithm 2 achieves average consensus in the arithmetic sense.

Although the compensated local clock $\check{x}_i(t)$ in (24) achieves synchronization for both the rate and offset, it may experience a step jump at the time the compensation starts. The jump can even be negative, causing temporary time reversal. So we modify (24) to:

$$\hat{x}_i(t) = \begin{cases} x_i(t), & t \leq t_i(\tau) \\ \{\exp(-\eta_i(d))(x_i(t) - \tau) + \tau \\ -(1 - \exp(-\mu_i(x_i(t) - \tau)))\gamma_i(d)\}, & t > t_i(\tau) \end{cases} \tag{26}$$

for some constant $\mu_i > 0$.

The choice of μ_i determines how fast the offset compensation term $(1-\exp(-\mu_i(x_i(t)-\tau)))\gamma_i(d)$ decays to the desired value $\gamma_i(d)$. In practice, if we take $\exp(-5) \approx 0.0067 \approx 0$, then, choosing $\mu_i = 5/T_i$ for some $T_i > 0$ will ensure that the offset compensation term becomes roughly $\gamma_i(d)$ after T_i local minutes. The value of T_i also needs to be chosen appropriately to ensure that the compensated clock rate is sufficiently positive (at least not negative) for all time t. We have the following result:

Theorem 8. *Suppose the graph $\mathcal{G}$ is undirected and acyclic with diameter d and $\mu_i > 0$, $i \in \mathcal{V}$. Then, the compensated local clocks $\hat{x}_i(t)$ in (26) are continuous and*

$$\hat{x}_i(t) \to at + b,\ \ t \to \infty,\ \forall i \in \mathcal{V}, \tag{27}$$

where $a = \sqrt[n]{a_1 a_2 \ldots a_n}$ and $b = (\beta_1 + \beta_2 + \cdots \beta_n)/n$. By choosing suitably large $M > 0$ with $\exp(-M) \approx 0$ and choosing $\mu_i = M/T_i$ for some $T_i > 0$ will result in

$$\hat{x}_i(t) \approx at + b,\ \ \forall t \geq t_i(\tau) + T_i. \tag{28}$$

Moreover, by choosing

$$T_i \begin{cases} = \text{any positive number}, & \gamma_i(d) \leq 0 \\ \geq -\dfrac{M}{\epsilon}\gamma_i(d)\exp(\eta_i(d)), & \gamma_i(d) > 0 \end{cases} \tag{29}$$

for any $0 < \epsilon < 1$ will ensure that the compensated clock $\hat{x}_i(t)$ has a minimum rate of $(1-\epsilon)a$.

Proof. The continuity of $\hat{x}_i(t)$ is easily verified because $x_i(t_i(\tau)) = \tau$ which means that $\hat{x}_i(t) \to \tau$ as $t \to t_i(\tau)$ either from the left or from the right. Other than $t = t_i(\tau)$, it is clear that $\hat{x}_i(t)$ is continuous elsewhere. The asymptotic common rate and offset follow from Theorem 6 and that $\exp(-\mu(\hat{x}_i(t)-\tau)) \to 0$ as $t \to \infty$. The approximation result (28) follows accordingly.

To show the minimum clock rate of $\hat{x}_i(t)$ for the chosen T_i in (29), we use (26) to obtain its rate

$$\dot{\hat{x}}_i(t) = a - \mu_i a_i \exp(-\mu_i(x_i(t) - \tau))\gamma_i(d),\ \ t \geq t_i(\tau).$$

It is clear that when $\gamma_i(d) \leq 0$, $\dot{\hat{x}}_i(t) \geq a$, regardless of the choice of $T_i > 0$. When $\gamma_i(d) > 0$,

$$\begin{aligned} \dot{\hat{x}}_i(t) &\geq a - \mu_i a_i \gamma_i(d) \\ &= a(1 - \mu_i \exp(\eta_i(d))\gamma_i(d)) \geq a(1-\epsilon), \end{aligned}$$

after using $\mu_i = M/T_i$ and (29).

3.3. Complexity analysis

It is easy to verify that Algorithms 1–2 satisfy the complexity constraints (1)–(3). Indeed, in each iteration k, node i uses $(s_{j\to i}(k-1), \eta_{j\to i}(k-1), \gamma_{j\to i}(k-1))$ received from each neighbouring node j only once, and then transmits back the resulting $(s_{i\to j}(k), \eta_{i\to j}(k), \gamma_{i\to j}(k))$. The computational complexity of (17)–(21) is clearly $O(|\mathcal{N}_i|)$ for each node i. Finally, each node i only needs to store $(s_{i\to j}(k), \eta_{i\to j}(k), \gamma_{i\to j}(k))$ for each $j \in \mathcal{N}_i$ (other than $\eta_i(k)$ and $\gamma_i(k)$ as well as the temporary variables $s_i(k)$, $\tilde{\eta}_i(k)$ and $\tilde{\gamma}_i(k)$), so its complexity is also $O(|\mathcal{N}_i|)$. Combining with Theorems 3 and 6, the complexity of Algorithms 1–2 is $O(d|\mathcal{N}_i|)$ for each node i.

3.4. Application to cyclic network graphs

A main restriction of the proposed algorithm for distributed clock synchronization is that the WSN graph needs to be acyclic. For a cyclic graph, it is necessary to prune loop-forming edges so that the remaining graph becomes loop free. The resulting graph is called a *spanning tree*, a tree graph with all the nodes in the original graph. This can be done by applying the recently proposed distributed *depth-first search* DFS algorithm in Xie, Cai, Zhang, and Fu (2018), under the mild assumption that each node in the graph has a distinct numerical ID number. For convenience, this algorithm is listed below (Algorithm 0), which takes at most $d+1$ iterations to finish. Algorithm 0 uses a (fully distributed) max-consensus algorithm (Nejad, Attia, & Raisch, 2009) for finding the maximum value in a connected graph, which finishes in d iterations.

Algorithm 0 (DFS Algorithm for Spanning Tree)

- **Initialization**: Select a root node in $\mathcal{G}$ (by running max-consensus d iterations and setting the node with maximum ID number as the root node). Mark the root node as "visited" and all other nodes as "unvisited". Transmit a token from the root node to each of its neighbouring nodes.
- **Iterations** $k = 1, 2, \ldots$: For each node i, do the following:
  (1) If it does not receive the token, do nothing;
  (2) If it is "unvisited" and it receives the token from only *one* neighbour, then mark the node as "visited", and relay the token to all other neighbouring nodes without the "removal" mark, except the incoming edge (i.e., the edge where the token came from);
  (3) If it is "unvisited" and it receives the token from *multiple* neighbours, then mark the node as "visited", leave one (any one) incoming edge alone and mark all other the incoming edges as "removal", and then relay the token to all other neighbouring nodes without the "removal" mark, except the remaining incoming edge;
  (4) If it is already "visited" and it receives the token, mark the incoming edge as "removal" and do not relay the token further.

4. Properties and modifications

In this section, we discuss several properties and modifications of the proposed clock synchronization scheme.

4.1. Robustness against transmission delay and loss

Algorithm 2 is resilient to transmission delays due to the fact that each node uses the most recently received information from its neighbours to update its own information. A delayed arrival of information from neighbours will only delay the update and the convergence of the algorithm, without affecting the final consensus. To see this, for each edge e in $\mathcal{G}$, we denote its transmission time by $T(e)$ and allow it to be a positive integer only: $T(e) = 1$ (unit of time) if there is no delay and loss; $T(e) > 1$ if it involves $T(e)$ units of time delay or $T(e) - 1$ retransmissions or a combination of them. For a path p in $\mathcal{G}$, its *path transmission time* $T(p)$ is defined to be the time it takes to relay a packet along this path by adding up the transmission times of the edges. We have the following result.

Corollary 9. *Suppose there exists T (*maximum path transmission time*) such that $T(p) \le T$ for every path p in $\mathcal{G}$. Then, under the conditions of Theorem 3, Algorithm 1 achieves convergence after T iterations, i.e., $\eta_i(k) = \alpha_i - \bar{\alpha}$ for all $k \ge T$, $i \in \mathcal{V}$. Similarly, under the conditions of Theorem 6, Algorithm 2 achieves convergence after T iterations, i.e., $\gamma_i(k) = \beta_i - b$ for all $k \ge T$, $i \in \mathcal{V}$. In the case of no delay (i.e., $T = d$), we have $\eta_i(k) = \alpha_i - \bar{\alpha}$ and $\gamma_i(k) = \beta_i - b$ for all $k \ge d$, $i \in \mathcal{V}$.*

Proof. Consider any node i. All the nodes in the acyclic $\mathcal{G}$ can be organized as a tree with node i in the root. We see from (10)–(11) in Theorem 3 and (22) in Theorem 6 that if there is no delay and loss, then at each iteration k, the variables k hops away from node i are added to $s_i(k)$, $\tilde{\eta}_i(k)$ and $\tilde{\gamma}_i(k)$. If there is $T(e) > 1$ for some edge $e = (i_1, i_2)$ along some path p originated from node i (assuming node i_1 is closer to node i than node i_2), then the information in all the nodes along the path p starting from node i_2 will all be delayed, but they will be included in a later iteration. Since the maximum path transmission time is T, we conclude that after T iterations, $\eta_i(k) = \alpha_i - \bar{\alpha}$ and $\gamma_i(k) = \beta_i - b$ for all $k \ge T$, $i \in \mathcal{V}$.

Likewise, if packet loss happens, it is sufficient that retransmission happens and eventually the information arrives, which can be guaranteed by most communications protocols, e.g., Transmission Control Protocol (TCP), via Acknowledgement (ACK). This will result in transmission delay only and will not affect the final consensus.

4.2. Asynchronous implementation

Although Algorithms 1–2 are written as a synchronous algorithm (i.e, every node uses the same time index k and updates at the same time), this is purely for convenience. The implementation can be completely asynchronous without degradation to the synchronization result. That is, no synchronicity is needed among the nodes in the sensor network. More specifically, each node simply needs to wait till it receives new information from its neighbours, then update its own information and transmit the updated information to its neighbours.

4.3. Regional clock synchronization

For a large-scale WSN, it is often sufficient and important for each node to be in sync with nodes over a sub-network around itself, the notion of *regional clock synchronization*. This has the obvious benefit that the latency over the sub-network is reduced, resulting in faster synchronization. A region of size δ for node i, denoted by $\mathcal{G}_i^\delta$, is the sub-network with all the nodes at most δ hops away from node i, including node i. Similar to the analysis in the previous subsection, we recognize (19)–(20) in Theorem 6 that Algorithm 2 can be used to achieve this purpose by stopping after δ iterations, i.e., $\eta_i(\delta) = \alpha_i - \bar{\alpha}^{(i)}$ and $\gamma_i(\delta) = \beta_i - b^{(i)}$, where $\bar{\alpha}^{(i)}$ and $b^{(i)}$ are the averages of α_j and β_j, respectively, over $j \in \mathcal{G}_i^\delta$.

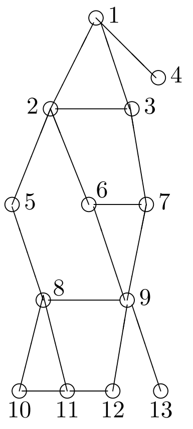
5. Illustrative examples

We illustrate our distributed clock synchronization algorithm using three examples. Comparisons are done with existing average-consensus based algorithms.

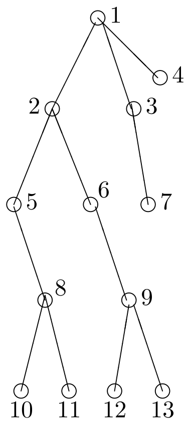
5.1. Example 1: Demonstration of Algorithms 1, 2 and 0

Considering the graph $\mathcal{G}$ in Fig. 1(a) with 13 nodes.

We first execute Algorithm 0 to generate a spanning tree. Using the max-consensus algorithm on $\mathcal{G}$ (using the negated labels) yields node 1 as the root node. The token is sent from node 1 to nodes 2, 3, 4 and so forth. The resulting spanning tree is in Fig. 1(b), with diameter $d = 6$ (a path from node 7 to node 13, for example).



(a) Original graph.



(b) Spanning tree.

Fig. 1. Graph of WSN.

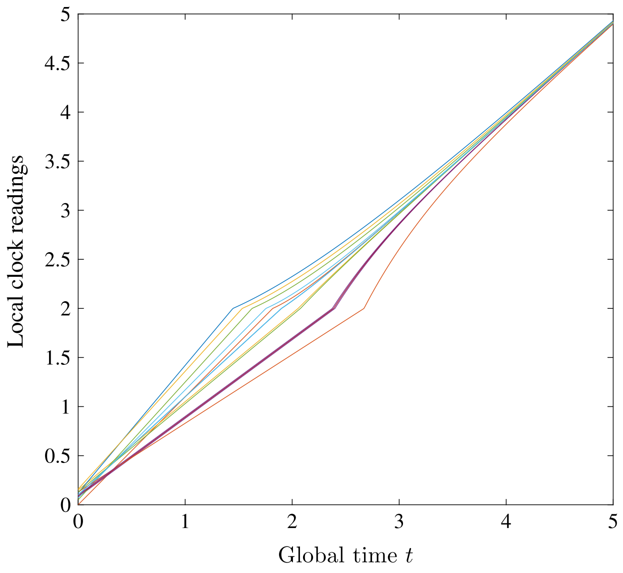


Fig. 2. Our algorithm: Before and after compensations.

Next, we apply Algorithms 1–2 to the graph in Fig. 1(b) with the local clock parameters $\{a_i\}$ and $\{b_i\}$ given by

$$a = [1\ 1.1\ 0.9\ 0.8\ 1.2\ 1.1\ 0.8\ 1.3\ 0.7\ 1.2\ 0.8\ 0.9\ 1]$$
$$b = [0.1\ 0\ 0.15\ 0.08\ 0.05\ 0.07\ 0.09\ 0.12\ 0.13\ 0.16\ 0.1\ 0.13\ 0.1].$$

We assume that execution time for each iteration k is 0.1 s. The local clock time $\tau = 2$ (minute) is used.

It follows that the global time $t_i(2)$ at which the local clock i reads $\tau = 2$ is given by $a_i t_i(2) + b_i = 2$, giving

$$\begin{aligned}&[t_1(2)\ t_2(2)\ \dots t_{13}(2)]\\ &= [1.9000\ 1.8182\ 2.0556\ 2.4000\ 1.6250\ 1.7545\ 2.3875\\ &\quad 1.4462\ 2.6714\ 1.5333\ 2.3750\ 2.0778\ 1.9000].\end{aligned} \tag{30}$$

Similarly, it is verified that the global time $t_i(1)$ at which the local clock i reads $\tau - 1 = 1$ minute is given by $a_i t_i(1) + b_i = 1$, which yields

$$\begin{aligned}&[t_1(1)\ t_2(1)\ \dots t_{13}(1)]\\ &= [0.9000\ 0.9091\ 0.9444\ 1.1500\ 0.7917\ 0.8455\ 1.1375\\ &\quad 0.6769\ 1.2429\ 0.7000\ 1.1250\ 0.9667\ 0.9000].\end{aligned} \tag{31}$$

We can compute $a = \sqrt[13]{a_1 a_2 \dots a_{13}} = 0.96826829$ and

$$\begin{aligned}&[\beta_1\ \beta_2\ \dots\ \beta_{13}]\\ &= [0.1603\ 0.2395\ 0.0097\ -0.3238\ 0.4266\ 0.3011\ -0.3117\\ &\quad 0.5997\ \ -0.5867\ 0.5153\ \ -0.2996\ \ -0.0118\ 0.1603],\end{aligned}$$

giving $b = 0.06755385$. The only information available to each node i is $x_i(t_j(1))$ and $x_i(t_j(2)), j \in \mathcal{N}$, e.g.,

$$\begin{aligned}&[x_1(t_2(1))\ x_1(t_3(1))\ x_1(t_4(1))] = [1.0091\ 1.0444\ 1.2500];\\ &[x_1(t_2(2))\ x_1(t_3(2))\ x_1(t_4(2))] = [1.9182\ 2.1556\ 2.5000].\end{aligned} \tag{32}$$

From the above, each node i computes $\delta_{ij} = \ln(x_i(t_j(2)) - x_i(t_j(1)))$, e.g., $[\delta_{12}\ \delta_{13}\ \delta_{14}] = \ln[0.9091\ 1.1112\ 1.2500]$.

Execute Algorithms 1 and 2 for 6 iterations each (or 1.2 s in total) with $\epsilon = 0.5, T_i \geq 2$ and $M = 5$. The compensated local clocks $\hat{x}_i(t)$ in (26) are shown in Fig. 2. We see from Fig. 2 that before the local clocks' readings reach $\tau = 2$, each clock runs at its own rate with its own offset. After they reach $\tau = 2$, they start to converge. At the global time $t = 5$, all the local clocks have converged to the common rate and common offset.

Next, we test the algorithms' resilience to transmission delay and packet loss. Assume that 1) a unit of delay (one iteration time) occurs for the transmission of $(x_{1\to 2}(3), s_{1\to 2}(3))$; 2) a packet loss occurs for $(x_{3\to 7}(4), s_{3\to 7}(4))$. It is verified that both algorithms take at most 7 iterations to converge, totalling 1.4 s, which is only 0.2 s slower than in the case without delay or packet loss. Also, due to the choice of $M = 5$, the achieved accuracy at $t = 5$ s is about $\exp(-5) \approx 0.0067$ or 0.67%.

Note that since the slowest local clock (clock 9) reaches $\tau = 2$ at global time $t = 2.6714$ (minutes), Algorithms 1–2 can be executed only after $t = 2.6714$. The compensations for the local clocks can be done only after the algorithms are terminated. This means that the un-compensated local clock readings are recorded and these readings are revised after the algorithms are terminated.

5.2. Comparison with Laplacian matrix-based algorithms

Now we compare the simulation results above with Laplacian matrix-based consensus algorithms for clock synchronization. It is well known that, for a Laplacian matrix-based algorithm, convergence is asymptotic only and the convergence rate depends on the ratio of the second and largest eigenvalues of the Laplacian matrix, which in turn depends highly on the network topology (see, e.g., Li et al., 2011). In contrast, Theorem 3 shows that at iteration k, all the nodes in the sub-graph within k hops away from node i are used to derive the averaged clock rate; see (10)–(11). Hence, the convergence rate depends on how many nodes are absorbed in iteration k, and convergence completes when $k = d$ (the graph diameter). Similar comment applies to the clock offset synchronization.

The most representative Laplacian matrix-based algorithm was proposed in Carli et al. (2011) with iterative dynamics:

$$z_i(h+1) = z_i(h) + u_i(h) \tag{33}$$

where $z_i(h) = [z_i(hT_0)'\ z_i(hT_0)'']^T$ with $z_i'(hT_0)$ represents the compensated (or estimated) local clock time for node i at time hT_0, and $z_i''(hT_0)$ represents the corresponding compensated (estimated) oscillation period (i.e., the reciprocal of clock rate), T_0 is the sampling period and $h = 0, 1, 2, \dots$. The control $u_i(h)$ is given by

$$u_i(h) = -\sum_{j\in\mathcal{N}_i} K_{ij} \left[\begin{matrix} 0.5 \\ 1/(T_0 \max\limits_i a_i) \end{matrix}\right] (z_j(hT_0)' - z_i(hT_0)')$$

and K_{ij} are elements of a Laplacian matrix K given by

$$K_{ij} = \begin{cases} \dfrac{1}{\max\{|\mathcal{N}_i|, |\mathcal{N}_j|\}}, & \text{if } i \neq j, (i,j) \in \mathcal{E} \\ -\sum\limits_{j \neq i} K_{ij} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

The initial conditions are chosen as $z_i'(0) = x_i(0), z_i''(0) = 1$ (the "nominal" clock rate). Between the update times hT_0 and $(h+1)T_0$, the compensated local clocks are

$$z_i'(t) = z_i''(hT_0)(x_i(t) - x(hT_0)) + z_i'(hT_0). \tag{34}$$

The above algorithm requires synchronous sampling, which is impractical. A modification is provided in Carli et al. (2011) to allow asynchronous implementation, with some performance degradation. For simplicity, we will do comparison only with the synchronous implementation. Applying the above to Fig. 1(a), we get

$$K = \begin{bmatrix} \frac{11}{12} & -\frac{1}{4} & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & -1 & -\frac{1}{4} & 0 & -\frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{4} & \frac{11}{12} & 0 & 0 & 0 & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & 0 & \frac{1}{2} & 0 & 0 & -\frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & 0 & 0 & \frac{47}{60} & -\frac{1}{3} & 0 & -\frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & \frac{13}{15} & 0 & -\frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{4} & 0 & 0 & \frac{19}{20} & -\frac{1}{5} & -\frac{1}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & -1 & 0 & 0 & -\frac{1}{5} & -\frac{1}{5} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{7}{12} & -\frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{4} & 0 & -\frac{1}{3} & \frac{11}{12} & -\frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{5} & 0 & -\frac{1}{3} & \frac{8}{15} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{5} & 0 & 0 & 0 & \frac{1}{5} \end{bmatrix}.$$

We implement (33)–(34) starting from $t = 2$ (as the initial time) and take $T_0 = 0.05$. Note that $\max_i a_i = 1.3$. Running the algorithm for 60 iterations, we obtain the results in Fig. 3. This value of T_0 is chosen such that the accuracy and convergence time are on par with Fig. 2 for the transient period (2–5 min). But we see that the algorithm in Carli et al. (2011) requires a lot more iterations and has serious discontinuity and time reversal behaviours.

We also comment that the relative accuracy of the compensated clock rate in Fig. 3 is 4.88% after 60 iteration. To improve the accuracy to 0.67% (the same as the proposed scheme), 160 iterations are needed. With $T_0 = 0.05$ s, this requires settling time of 8 min. Making T_0 shorter would become impractical for most wireless sensors. In addition, even 8 min of settling time is acceptable, this would require constant communications and computations (once every 0.05 s), which means that a main operating mode – sleep mode – cannot be used. In contrast, our scheme needs $2d = 12$ iterations only and settling time of about 3 min (or shorter if larger μ_i values are used). Also recall that the above comparison is with the synchronous implementation (Carli et al., 2011).

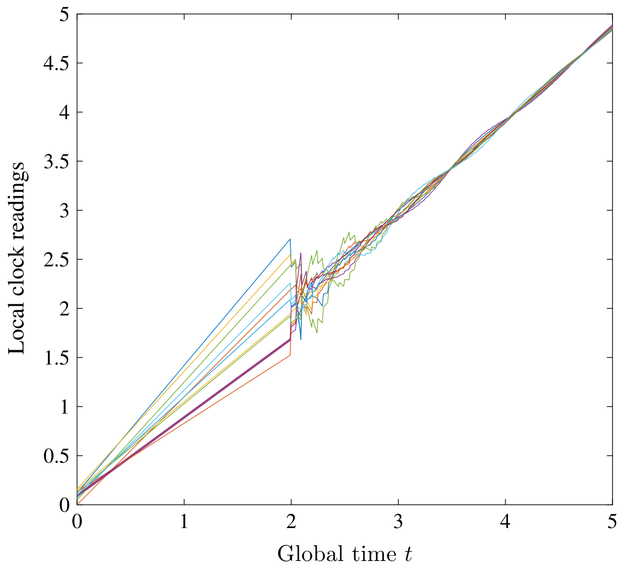


Fig. 3. Algorithm in Carli and Zampieri (2014): Before and after compensations.

5.3. Example 2: scalability to large WSNs

To demonstrate the scalability of the proposed algorithm, we consider a network with $n = 500$ nodes. It is reasonable to expect its diameter to be around $d = 10$ (or we can use $d = 10$ for regional clock synchronization). It is assumed that each local clock drifts up to 8 (i.e., ± 4) seconds every 24 h. (Note that this figure is on par with the accuracy of a typical 1 μW clock for wireless sensors, which is about 10^{-4} parts-per-million (PPM), or 0.01%; see, e.g., Sadler, 2005.) To guarantee a maximum error of 0.01 (i.e., ± 0.005) second, it is sufficient to re-synchronize the clocks every 2 min. Assuming each iteration of Algorithm 1 or 2 takes 0.05 s, then 1 s (for every 2 min) will be enough to synchronize all the clocks.

To give a specific example, we consider a WSN testbed studied in Schenato and Fiorentin (2011) where each sensor node uses an external crystal oscillator (ECO) running at 32768 Hz during the idle state. The corresponding clock resolution is 1 tick = 1/32768 Hz = 30.5 μs. Assuming that the maximum clock rate drift is 10^{-4} PPM as above, then the maximum clock offset is $0.0001 \times 2 \times 60 = 0.012$ s, which is 392 ticks. Suppose we do not do the rate synchronization and simply do the offset synchronization every 2 min, i.e., run Algorithm 2 only once every 2 min. Then, after each re-synchronization, the offset error is reduced back to $\exp(-5) \times 392 = 2.64 < 3$ ticks, assuming that $M = 5$ and μ_i are chosen to have settling time less than 2 min. If rate synchronization is also done, then this compensates about 90% of the clock drift, i.e., the maximum offset error is reduced down to 39.2 ticks after each re-synchronization. Together with Algorithm 1, the clock offset error is reduced down to 0.264 tick.

5.4. Example 3: resilience to measurement errors

To test the robustness of the proposed scheme against measurement errors, we revisit Example 1 by adding measurement noises and quantization errors. We first claim that, *under the assumption that the measurement errors are modelled by zero-mean random variables with variance σ^2, the induced error variances in the compensated clock rate and offset are approximately $2\sigma^2/n$ in both cases*, resulting in great robustness.

To see the above claim, we consider node i and node $j \in \mathcal{N}_i$. Recall that the global time $t_j(\tau)$ at which the local clock j announces the τ-th minute is given by $a_j t_j(\tau) + b_j = \tau$ in the noise-free case. We now modify this to

$$a_j t_j(\tau) + b_j = \tau + e_j(\tau) \tag{35}$$

with the quantity $e_j(\tau)$ including both measurement noise and quantization error at time τ. Denote the induced error in $t_j(\tau)$ as $\Delta t_j(\tau)$. Taking the approximation of $a_j \approx 1$, we get $\Delta t_j(\tau) \approx e_j(\tau)$. Similarly, the local clock $x_i(t) = a_i t + b_i$ is modified to

$$x_i(t) = a_i t + b_i + e_i(t). \tag{36}$$

Also taking the approximation of $a_i \approx 1$, we get

$$\Delta x_i(t_j(\tau)) \approx \Delta t_j(\tau) + e_i(t_j(\tau)) \approx e_j(\tau) + e_i(\tau).$$

Following (3) and using the further approximation that $x_i(t_j(\tau)) - x_i(t_j(\tau-1)) \approx 1$ and the fact that $\ln(1+x) \approx x$ for small x, the induced error in $\delta_{ij}(\tau)$ becomes

$$\Delta\delta_{ij}(\tau) \approx e_j(\tau) + e_i(\tau) - e_j(\tau-1) - e_i(\tau-1).$$

Under the assumption that $e_i(\tau)$ is independent of τ and i with zero mean and variance σ^2, $\Delta\delta_{ij}(\tau)$ is approximately zero mean with variance of $4\sigma^2$. Using (4), this is equivalent to each δ_i having a measurement error of zero mean and variance of $2\sigma^2$. Consequently, the induced error in the arithmetic mean $\bar{\alpha}$ of α_i, denoted by $\Delta\bar{\alpha}$ is approximately zero mean with variance of $\frac{2}{n}\sigma^2$. Since the compensated clock rate is also approximately equal to 1, we get our conclusion that the induced error in the compensated clock rate is approximately a zero mean with variance of $\frac{2}{n}\sigma^2$. The verification for the compensated offset is similar and thus omitted.

Taking the ECO of 32 768 Hz as an example, we assume $e_i(\tau)$ to be a random binary value of ± 0.5 tick $= 10.25\ \mu$s. These values will be added to $t_i(2)$, $t_i(1)$, $x_i(t_j(2))$ and $x_i(t_j(1))$ in (30)–(32). After running Algorithms 1–2, the newly computed values for α and b are $\alpha \approx 0.968273$ and $b = 0.067556$, respectively. Compared with the previous values in Example 1, the effect of measurement errors is indeed negligible. It is verified that the error variances match the predicted value of σ^2/n. We further assume that the readings of $x_i(t_j(\tau))$ are quantized by truncating to 2 decimal points, resulting in

$$[x_1(t_2(1))\, x_1(t_3(1))\, x_1(t_4(1))] = [1.01\ 1.04\ 1.25];$$
$$[x_1(t_2(2))\, x_1(t_3(2))\, x_1(t_4(2))] = [1.92\ 2.16\ 2.50].$$

Even in this case, the new values are $\alpha = 0.9682$ and $b = 0.067$, respectively, which are negligibly different.

6. Conclusions

A new *fully distributed* clock synchronization algorithm has been proposed for WSNs. The algorithm is developed based on a finite average consensus concept. For WSNs with an acyclic graph, the algorithm is shown to converge in only d iterations for clock rate synchronization and then d iterations for offset synchronization, where d is the graph diameter. Many nice properties of the algorithm have been studied, including low complexities, robustness against transmission adversaries and asynchronous implementability. Modification of the algorithm is presented for the regional clock synchronization problem.

References

Ahmad, A., Zennaro, D., Serpedin, E., & Vangelista, L. (2012). A cactor graph approach to clock offset estimation in wireless sensor networks. *IEEE Transactions on Information Theory*, *58*(7), 4244–4260.

Bolognani, S., Carli, R., Lovisari, E., & Zampieri, S. (2016). A randomized linear algorithm for clock synchronization in multi-agent systems. *IEEE Transactions on Automatic Control*, *61*(7), 1711–1726.

Carli, R., Chiuso, A., Schenato, L., & Zampieri, S. (2011). Optimal synchronization for networks of noisy double integrators. *IEEE Transactions on Automatic Control*, *56*(5), 1146–1152.

Carli, R., D'Elia, E., & Zampieri, S. (2011). A PI controller based on asymmetric gossip communications for clocks synchronization in wireless sensors networks. In *Proceedings of 50th IEEE conference on decision and control and European control conference* (pp. 7512–7517).

Carli, R., & Zampieri, S. (2014). Network clock synchronization based on the second-order linear consensus algorithm. *IEEE Transactions on Automatic Control*, *59*(2), 409–422.

Chaudhari, Q., Serpedin, E., & Qaraqe, K. (2008). On maximum likelihood estimation of clock offset and skew in networks with exponential delays. *IEEE Transactions on Signal Processing*, *56*(4), 1685–1697.

Chen, J., Yu, Q., Zhang, Y., Chen, H.-H., & Sun, Y. (2010). Feedback-based clock synchronization in wireless wireless sensor networks: a control theoretic approach. *IEEE Transactions on Vehicular Technology*, *59*(6), 2963–2973.

Cortés, J. (2006). Finite-time convergent gradient flows with applications to network consensus. *Automatica*, *42*(11), 1993–2000.

Dai, H., & Han, R. (2004). Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, *8*(1), 125–139.

Du, J., & Wu, Y.-C. (2013). Distributed clock skew and offset estimation in wireless sensor networks: asynchronous algorithm and convergence analysis. *IEEE Transactions on Wireless Communications*, *12*(11), 5908–5917.

El Khediri, S., Nasri, N., Samet, M., Wei, A., & Kachouri, A. (2012). Analysis study of time synchronization protocols in wireless wireless sensor networks. *International Journal of Distributed and Parallel Systems*, *3*(3), 155–165.

Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. In *Proc. fifth symposium on operating systems design and implementation, Vol. 36* (pp. 147–163).

Ferrari, F., Zimmerling, M., Thiele, L., & Saukh, O. (2011). Efficient network flooding and time synchronization with glossy. In *Proc. 10th int. conf. information processing in wireless sensor networks* (pp. 73–84).

Ganeriwal, S., Kumar, R., & Srivastava, M. B. (2003). Timing-sync protocol for sensor networks. In *Proc. 1st ACM conf. embedded networked sensor sys.* (pp. 138–149).

Hendrickx, J. M., et al. (2004). Graph diameter, eigenvalues, and minimum-time consensus. *Automatica*, *50*, 635–640.

Kadowaki, Y., & Ishii, H. (2015). Event-based distributed clock synchronization for wireless sensor networks. *IEEE Transactions on Automatic Control*, *60*(8), 2266–2271.

Kashyap, A., Basar, T., & Srikant, R. (2007). Quantized consensus. *Automatica*, *43*, 1192–1203.

Leng, M., & Wu, Y.-C. (2011). Distributed clock synchronization for wireless sensor networks using belief propagation. *IEEE Transactions on Signal Processing*, *59*(11), 5404–5414.

Leva, A., Terraneo, F., Rinaldi, L., Papadopoulos, A. V., & Maggio, M. (2016). High-precision low-power wireless nodes' synchronization via decentralized control. *IEEE Transactions on Control Systems Technology*, *24*(4), 1279–1293.

Li, T., Fu, M., Xie, L., & Zhang, J. (2011). Distributed consensus with limited communication data rate. *IEEE Transactions on Automatic Control*, *56*(2), 279–292.

Luo, B., & Wu, Y. C. (2013). Distributed clock parameters tracking in wireless sensor network. *IEEE Transactions on Wireless Communications*, *12*(12), 6464–6475.

Maróti, M., Kusy, B., Simon, G., & Lédeczi, A. (2004). The flooding time synchronization protocol. In *Proc. 2nd int. conf. embedded networked sensor systems* (pp. 39–49). New York, NY, USA: ACM.

Mills, D. (1991). Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, *39*(10), 1482–1493.

Mock, M., Frings, R., Nett, E., & Trikaliotis, S. (2000). Continuous clock synchronization in wireless real-time applications. In *Proc. 19th IEEE symposium on reliable distributed systems* (pp. 125–133).

Nejad, B. M., Attia, S. A., & Raisch, J. (2009). Max-consensus in a max-plus arithmetic setting: the case of fixed communication topologies. In *XXII international symposium on information, communication and automation technologies*.

Noh, K.-L., Serpedin, E., & Qaraqe, K. (2008). A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization. *IEEE Transactions on Wireless Communications*, *7*(9), 3318–3322.

Ping, S. (2003). Delay measurement time synchronization for wireless sensor networks, Intel Research, IRB-TR-03-013, June.

Ren, F., Lin, C., & Liu, F. (2008). Self-correcting time synchronization using reference broadcast in wireless wireless sensor network. *IEEE Transactions on Wireless Communications*, *15*(4), 79–85.

Rhee, I.-K., Lee, J., Kim, J., Serpedin, E., & Wu, Y.-C. (2009). Clock synchronization in wireless wireless sensor networks: An overview. *Sensors*, *9*(1), 56–85.

Sadler, B. M. (2005). Fundamentals of energy-constrained wireless sensor network systems. *IEEE Transactions on Aerosp. Electron. Syst.*, *20*(8), 17–35.

Sarvghadi, M. A., & Wan, T.-C. (2014). Overview of time synchronization protocols in wireless wireless sensor networks. In *2nd international conference on electronic design*.

Schenato, L., & Fiorentin, F. (2011). Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, *47*(9), 1878–1886.

Sichitiu, M. L., & Simple, V. C. (2003). Accurate time synchronization for wireless sensor networks. In *Proc. IEEE wireless commun. and networking*.

Simeone, O., & Spagnolini, U. (2007). Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators, U. *Journal of Wireless Communications and Networking*, *2007*(1), 1–13.

Solis, R., Borkar, V. S., & Kumar, P. R. (2006). A new distributed time synchronization protocol for multihop wireless networks. In *IEEE conference on decision and control*.

Sundararaman, B., Buy, U., & Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: A survey. *Ad-Hoc Networking*, *3*(3), 281–323.

Van Greunen, J., & Rabaey, J. (2003). Lightweight time synchronization for sensor networks. In *Proc. 2nd ACM intertional workshop on wireless sensor networks and apps.* (pp. 11–19).

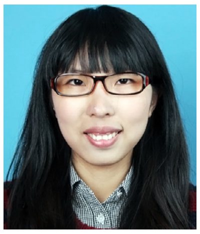
Xiao, L., & Boyd, S. (2004). Fast linear iterations for distributed averaging. *Systems & Control Letters*, *53*, 65–78.

Xie, K., Cai, Q., Zhang, Z., & Fu, M. (2018). A fast converging distributed algorithm for weighted average consensus. *IEEE Transactions on Automatic Control*, submitted for publication.

Yoon, S., Veerarittiphan, C., & Sichitiu, M. L. (2007). Tiny-sync: Tight time synchronization for wireless wireless sensor networks. *ACM Transactions on Wireless Sensor Networks*, *3*, 2.

Zennaro, D., et al. (2013). Network-wide clock synchronization via message passing with exponentially distributed link delays. *IEEE Transactions on Communications*, *61*(5), 2012–2024.

Kan Xie was born in Hubei, China. He received the M.S. degree in software engineering from the South China University of Technology, Guangzhou, China, in 2009. Currently, he is pursuing the Ph.D. degree in intelligent signal and information processing at the Guangdong University of Technology, Guangzhou, China. His research interests include machine learning, nonnegative signal processing, blind signal processing, and biomedical signal.



Qianqian Cai received the B.S. degree in environmental engineering from the University of Shanghai for Science and Technology, Shanghai, China, in 2011, and the Ph.D. degree in engineering from the University of Newcastle, Newcastle, Australia, in 2016. She is currently doing post-doctoral research in the School of Automation, Guangdong University of Technology, Guangdong, China. Her interests include water pollution control engineering, environmental monitoring, automation and adaptive neural fuzzy inference systems.

Minyue Fu received his Bachelor's Degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 1982, and M.S. and Ph.D. degrees in electrical engineering from the University of Wisconsin-Madison in 1983 and 1987, respectively. From 1983 to 1987, he held a teaching assistantship and a research assistantship at the University of Wisconsin-Madison. From 1987 to 1989, he served as an Assistant Professor in the Department of Electrical and Computer Engineering, Wayne State University, Detroit, Michigan. He joined the Department of Electrical and Computer Engineering, University of Newcastle, Australia, in 1989. Currently, he is a Chair Professor in Electrical Engineering and Head of the School of Electrical Engineering and Computer Science. In addition, he was a Visiting Associate Professor at the University of Iowa in 1995–1996, and a Visiting Professor at Nanyang Technological University, Singapore in 2002, Chang Jiang Professor at Shandong University in 2007–2010, a Qian-ren Scholar at Zhejiang University and Guangdong University of Technology, China. He is a Fellow of the IEEE. His main research interests include control systems, signal processing and communications. He has been an Associate Editor for the IEEE Transactions on Automatic Control, IEEE Transactions on Signal Processing, Automatica and Journal of Optimization and Engineering.