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

ARTICLE INFO

Article history:
Received 21 May 2016
Received in revised form 18 September 2017
Accepted 13 December 2017

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

ABSTRACT

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/

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