Clock Synchronization Protocol for Wireless Sensor Networks with Bounded Communication Delays

Emanuele Garone [a], Andrea Gasparri [b], Francesco Lamonaca [c]

[a] *Faculty of Applied Science, Control and Systems Analysis Department, Université Libre de Bruxelles, 1050 Brussels, Belgium*

[b] *Department of Engineering, Roma Tre University, Via della Vasca Navale 79, 00146 Rome, Italy*

[c] *Dept. of Computer Science, Modeling, Electronic and Systems, University of Calabria, Ponte P. Bucci, 87040, Arcavacata di Rende, Italy*

Abstract

In this paper, we address the clock synchronization problem for wireless sensor networks. In particular, we consider a wireless sensor network where nodes are equipped with a local clock and communicate in order to achieve a common sense of time. The proposed approach consists of two asynchronous consensus algorithms, the first of which synchronizes the clocks frequency and the second of which synchronizes the clocks offset. This work advances the state of the art by providing robustness against bounded communication delays. A theoretical characterization of the algorithm properties is provided. Simulations and experimental results are presented to corroborate the theoretical findings and show the effectiveness of the proposed algorithm.

Key words: Asynchronous Consensus Algorithm, Clock Synchronization, Noise Robustness, Delay Measurements, Wireless Sensor Networks

1 Introduction

A Wireless Sensor Network (WSN) consists of a collection of nodes deployed within an environment to perform a given task. Each node is typically equipped with a radio transceiver, a micro-controller and a set of sensors. Nodes collaborate in order to reach a common goal. WSNs are at the forefront of emerging technologies due to the recent advances in Microelectromechanical Systems (MEMSs). The inherent multidisciplinary nature of WSNs has attracted scientists coming from different research areas from networking to robotics. Their application ranges from surveillance and coverage [1,2], structural health monitoring [3,4], and industrial process control [5,6] to emergency response [7,8] and mobile target tracking [9,10]. Most of these applications require basic services such as self-localization [11,12], time synchronization [13,14], and topology control [15,16]. However, the distributed nature and the limited hardware capabilities of WSNs make the development of these applications and related services particularly challenging.

In this work, we advance the state of the art by providing a solution to the clock synchronization problem in the presence of bounded communication delays. Inspired by the Average TimeSynch Protocol introduced in [14], we propose a novel synchronization protocol, denoted as Robust Average TimeSynch (RoATS), to adjust both the nodes' clock frequency and clock offset in a robust way with respect to bounded communication delays. A preliminary version of the RoATS was presented in [17]. This paper considerably improves that preliminary results in several directions. In particular, the drift and offset compensation are now considered as two independent and asynchronous processes. As a result, the convergence analysis has been revised and more meaningful theoretical bounds have been obtained. Finally, experiments have been carried out to corroborate the theoretical finding in a real-world environment.

⋆ This work was partially supported by the Italian grant FIRB "Futuro in Ricerca", project NECTAR, code RBFR08QWUV, funded by the Italian Ministry of Research and Education (MIUR) and by the Italian Ministry of Economic Development by means of the RIDITT: "Italian Network for Innovation and Technology Transfer to Enterprises", under the project "DI.TR.IM.MIS".

Email addresses: `egarone@ulb.ac.be` (Emanuele Garone), `gasparri@dia.uniroma3.it` (Andrea Gasparri), `flamonaca@deis.unical.it` (Francesco Lamonaca).

2 Related Work

Clock synchronization is an important problem in the context of distributed systems. This problem has become particularly relevant with the introduction of the Internet, as large networks of connected computers became more and more common. In this context, the most famous protocol is the Network Time Protocol (NTP) introduced in [18]. NTP was designed for large-scale networks with a rather static topology (such as the Internet). Nodes are externally synchronized to a global reference time that is injected into the network at many places via a set of master nodes. Master nodes are synchronized out of band, for example via GPS, and diffuse the notion of time to the other nodes by following a hierarchical scheme. Major limitations concerning the application of NTP to WSNs are the highly dynamic nature of the network topology, the limited bandwidth, and the necessity of a completely decentralized architecture to ensure robustness and flexibility. In the past years, several algorithms have been designed to deal specifically with the typical WSNs requirements, such as low energy consumption, bandwidth constraints and long-term operation. Several surveys concerning the clock synchronization problem in WSNs can be found in the literature. Among the others, it is worth mentioning [19], where a comparison of different synchronization protocols is carried out on the basis of a palette of performance indexes, e.g., precision, accuracy, cost, and complexity. A more recent survey can be found in [20], where the latest advances in the field of clock synchronization of WSNs are reported from a signal processing perspective. Apart from the seminal approaches such as the Reference-Broadcast Synchronization (RBS) [21] and the Timing-sync Protocol for Sensor Networks (TPSN) [22], relatively few protocols taking into account noisy measurements and delays have been proposed. In [23], the authors introduce a protocol to synchronize a network of controlled discrete-time double integrators which are nonidentical, with unknown model parameters and subject to additive measurement and process noise. In [24], by assuming the dynamic nature of the network to be modeled as a Markov chain, the authors propose a distributed algorithm for the estimation of scalar parameters from noisy relative measurements. In particular, they prove the estimates to be mean square convergent under fairly weak assumptions. In [25], the joint Maximum-Likelihood Estimation (MLE) of clock offset and skew is introduced assuming an exponential delay model. In [26], the authors derive three clock-synchronization algorithms for WSNs under unknown delays. In [27], a clock synchronization algorithm, called the Iterative Gaussian mixture Kalman particle filter (IGMKPF) is introduced. Briefly, this combines the Gaussian mixture Kalman particle filter (GMKPF) with an iterative noise density estimation procedure to achieve robust performance in the presence of unknown network delay distributions. Recently, the design of completely decentralized synchronization algorithms based on the consensus approach has gained momentum. Along this line, in [28] an algorithm based on a PI-like consensus protocol has been introduced, where the proportional (P) part compensates the different clock frequencies while the integral part (I) eliminates the different clock offsets. In [14], a distributed clock synchronization protocol, referred to as Average TimeSync (ATS), has been introduced. This protocol, which will be detailed in Section 4, is based on the cascade of two consensus algorithms aiming at synchronizing the clock drift and offset, respectively. In [29], a consensus-based protocol aiming at reducing the clock error between geographically closely located nodes has been proposed. In [13], the clock synchronization problem for event-driven measurement applications is addressed. In particular, the authors propose a consensus-based protocol which allows to achieve high accuracy in the area where an event is detected and ensure long network lifetime.

Compared to the state of the art, this paper provides a novel robust protocol with provable guarantees on the synchronization accuracy against bounded communication delays. More specifically, two asynchronous consensus-based protocols for the synchronization of the clocks frequency and offset are introduced. Their convergence properties are theoretically characterized and experimentally validated by means of a WSN composed of TelosB nodes [30].

3 Problem Statement

Consider a wireless sensor network composed of N nodes and assume that the network topology is described by means of an undirected connected graph $\mathcal{G} = (V, E)$, where $V = \{1, \ldots, N\}$ is the set of vertices representing the sensor nodes and $E = \{(i,\, j)\}$ is the set of edges describing the point-to-point channel availability. Namely, an edge $(i,\, j)$ exists if node i can transmit a packet to node j. Note that, since the network topology is assumed undirected the existence of an edge $(i,\, j)$ implies the existence of the edge $(j,\, i)$. Communication between pairs of nodes is assumed to be asynchronous.

Each node i is equipped with a local *hardware* clock τ_i defined as:

$$\tau_i(t) = \alpha_i\, t + \beta_i, \tag{1}$$

where $\alpha_i \in [\alpha_{\min},\, \alpha_{\max}]$ is the local clock frequency and β_i is the local clock offset. Coefficients $(\alpha_i,\, \beta_i)$ may differ for each node due to construction imperfections and different operational conditions, e.g., different temperatures for the quartz oscillators. Thus, in absence of corrections, the notion of time among the nodes may quickly diverge. To address this issue, each node is provided with a tunable *software* clock $\hat{\tau}_i(t)$ defined as:

$$\hat{\tau}_i(t) = \hat{\alpha}_i(t)\, \tau_i(t) + \hat{o}_i(t), \tag{2}$$

where $\hat{\alpha}_i(t)$ and $\hat{o}_i(t)$ are scalar parameters that can be used to adjust the i-th clock frequency and offset, respectively.

The objective of the synchronization problem is to adjust these parameters to eventually achieve a common sense of time in the software clock of the nodes, that is:

$$\lim_{t\to\infty} [\hat{\tau}_i(t) - \hat{\tau}_j(t)] = 0, \quad \forall i,\, j\, \in V. \tag{3}$$

Clearly, perfect synchronization is achievable only in the ideal case of instantaneous transmission. Differently, in a more realistic scenario where (random) bounded transmission delays may occur, the synchronization objective is to ensure that the difference between the nodes' clock remains bounded.

4 The ATS Protocol

In this section, for the reader's convenience and for the sake of comparison, the main aspects of the ATS protocol proposed in [14] are briefly reviewed. In the ATS, for each update k only one directed arc $e_k = (j, i)$ of $\mathcal{G}$ is involved. This implies that a single packet from node j to node i is sent. This packet contains the tuple $(\text{id}_j, \hat{\alpha}_j, \hat{o}_j, \tau_j)$, where id_j is the ID of the j-th node, $\hat{\alpha}_j = \hat{\alpha}_j(k-1) = \hat{\alpha}_j(t_k)$ and $\hat{o}_j = \hat{o}_j(k-1) = \hat{o}_j(t_k)$ are the parameters of its local software clock at time t_k, and $\tau_j = \tau_j(t_k)$ is the hardware timestamp of the packet, i.e. the value of the hardware clock of node j at the moment the message is sent. Upon packet reception, node i performs a timestamp of its hardware local clock in a variable $\tau_{ij} = \tau_i(t_k)$. We reiterate that in this context packet transmission is assumed to be ideal and thus no delay occurs.

At this point node i executes the local synchronization procedure consisting of three steps:

1 **Drift estimation:** The i-th node computes the k-th drift estimate $\alpha_{ij}(k) = \alpha_j/\alpha_i$ comparing τ_i and τ_j in two different time instants and evaluating the relative slopes. This requires that node i stores two variables $\tau_j(t_{k'})$, $\tau_{ij}(t_{k'})$ for each neighbor j, with $t_{k'}$ the last time this exchange occurred. It follows that:

$$\alpha_{ij}(k) = \frac{\tau_j(t_k) - \tau_j(t_{k'})}{\tau_i(t_k) - \tau_{ij}(t_{k'})}. \tag{4}$$

2 **Drift compensation**: $\hat{\alpha}_i$ is updated by using the drift estimate $\alpha_{ij}(t_k)$ as follows:

$$\hat{\alpha}_i(k) = \rho_v \hat{\alpha}_i(k-1) + [1-\rho_v]\alpha_{ij}(k)\hat{\alpha}_i(k-1), \tag{5}$$

where $\rho_v \in (0, 1)$ is a design parameter.

3 **Offset compensation:** $\hat{o}_i$ is updated as follows:

$$\hat{o}_i(k) = \hat{o}_i(k-1) + [1-\rho_o][\hat{\tau}_j(t_k) - \hat{\tau}_i(t_k)] \tag{6}$$

where $\rho_o \in (0, 1)$ is a design parameter.

The correction parameters are usually initialized as $\hat{\alpha}_i = 1$, $\hat{o}_i = 0$, with $i = 1, ..., N$. The following convergence result was proven in [14].

Theorem 1 *Consider a WSN running the ATS protocol. Assume α_i and β_i are constant $\forall\, i \in V$, the difference between two consecutive update times is bounded, i.e., $t_{k+1} - t_k \le \Delta_{t_{\max}}$, and there is no transmission delay. Then if there exists an integer Δk such that for any integer $k > 0$ the graph $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$ is strongly connected, all the software clocks will eventually synchronize:*

$$\lim_{t\to\infty} [\hat{\tau}_i(t) - \hat{\tau}_j(t)] = 0, \forall i, j \in V, \tag{7}$$

exponentially fast.

Remark 1 *As pointed out in [14] a low pass filter can be introduced to reduce the sensitivity of α_{ij}:*

$$\alpha_{ij}(k) = [1-\rho_l]\alpha_{ij}(k-1) + \rho_l \frac{\tau_j(t_k) - \tau_j(t_{k'})}{\tau_i(t_k) - \tau_{ij}(t_{k'})} \tag{8}$$

where $\rho_l \in (0, 1)$ is a design parameter. Low values of ρ_l are particularly useful to cope with high communication frequencies. The use of this low pass filter does not affect the theoretical properties of the algorithm.

Note that Theorem 1 holds under the assumption that an ideal communication channel is available, i.e., no transmission delays occur. As theoretically shown in Section 6.2 and experimentally demonstrated in Section 9.3, undesired behaviors may be experienced in a more realistic scenario where transmission delays occur, even if they are arbitrarily small. This paper addresses this issue by introducing a novel robust synchronization algorithm.

5 Robust ATS Algorithm - General Description

The Robust ATS (RoATS) proposed in this paper consists of two concurrent asynchronous algorithms:

(1) **A drift compensation protocol** concerning parameters $\hat{\alpha}_i, i = 1, ..., N$;
(2) **An offsets compensation protocol** concerning parameters $\hat{o}_i, i = 1, ..., N$.

A detailed description of these two protocols is given in the sequel.

6 Drift Compensation Protocol

At each update k of the drift compensation, an arc $e_k = (i, j)$ is selected. The two nodes involved in the update will:

1) send a first packet to the other node, receive the packet sent by the other, and compute their own *drift estimate*;
2) send a second packet to the other node to communicate the drift estimate, receive the packet sent by the other node, and perform a *symmetric drift compensation*;
3) *compensate* the effect of the *drift change* on the *offset*.

The main difference compared to the ATS algorithm is that the drift compensation is not carried out independently by each node. Instead, a further round of communication is introduced to achieve robustness against communication delays. Intuitively, this allows to recover an invariance property with respect to the parameters update as in the ideal case. A detailed description of these three steps is given hereafter.

6.1 Drift Estimation

In order to perform the k-th update, nodes i and j send each other a packet at time t_k^i and t_k^j, respectively. For the sake of simplicity, let us focus only on the actions performed by node i, as the same applies to agent j. The packet received by node i contains the tuple $(\mathrm{id}_j, \hat{\alpha}_j, \tau_j)$ where id_j is the ID of the j-th node, $\hat{\alpha}_j = \hat{\alpha}_j(t_k^j)$ is the drift parameter of its local software clock at time t_k^j, and $\tau_j = \tau_j(t_k^j)$ is the hardware timestamp of the packet performed by agent j. Upon packet reception, node i immediately stores the current value of its hardware local clock in a variable $\tau_{ij}(t_k^j) = \tau_i(t_k^j + \delta_k^i)$, where δ_k^i denotes the actual (unknown) bounded transmission delay.

As for the ATS algorithm, the information contained in the previous exchange of messages between nodes i and j is required to estimate the drift. Denote with k' the update when this exchange occurred. The following information about node j is then available to node i:

$$< \tau_j(t_{k'}^j),\, \tau_{ij}(t_{k'}^j),\, \tau_j(t_k^j),\, \tau_{ij}(t_k^j),\, \hat{\alpha}_j(t_k^j) >, \tag{9}$$

where:

$$\begin{aligned} \tau_j(t_{k'}^j) &= \alpha_j\, t_{k'}^j + \beta_j \\ \tau_{ij}(t_{k'}^j) &= \tau_i(t_{k'}^j + \delta_{k'}^i) = \alpha_i\, t_{k'}^j + \alpha_i\, \delta_{k'}^i + \beta_i \\ \tau_j(t_k^j) &= \alpha_j\, t_k^j + \beta_j \\ \tau_{ij}(t_k^j) &= \tau_i(t_k^j + \delta_k^i) = \alpha_i\, t_k^j + \alpha_i\, \delta_k^i + \beta_i \end{aligned} . \tag{10}$$

Node i then compute the (noisy) drift estimate as follows:

$$\begin{aligned} \alpha_{ij}(k) &= \frac{\tau_j(t_k^j) - \tau_j(t_{k'}^j)}{\tau_{ij}(t_k^j) - \tau_{ij}(t_{k'}^j)} \\ &= \frac{\alpha_j[t_k^j - t_{k'}^j]}{\alpha_i[t_k^j - t_{k'}^j] + \alpha_i[\delta_k^i - \delta_{k'}^i]} = \frac{\alpha_j}{\alpha_i}\,\Delta_{ij}(k) \end{aligned} , \tag{11}$$

where $\Delta_{ij}(k)$ is the unknown multiplicative delay factor defined as:

$$\Delta_{ij}(k) = \frac{1}{\left[1 + \frac{(\delta_k^i - \delta_{k'}^i)}{(t_k^j - t_{k'}^j)}\right]}. \tag{12}$$

Under the following realistic and reasonable assumptions:

- **Assumption 1**: The interval between the transmission of two consecutive packets for a couple of nodes is lower and upper bounded:
$$\Delta_{t_{\min}} \le t_k^i - t_{k'}^i \le \Delta_{t_{\max}},$$
for any update k involving node i;
- **Assumption 2**: There exists an upper bound $\delta_{\max}$ on the maximum delay for the reception of a packet:
$$0 \le \delta_k^i \le \delta_{\max}, \quad \forall\, k \ge 0;$$

the following bounds can be derived for the multiplicative delay factor $\Delta_{ij}(k)$:

$$\Delta_{\min} = \frac{1}{1 + \frac{\delta_{\max}}{\Delta_{t_{\min}}}} \le \Delta_{ij}(k) \le \frac{1}{1 - \frac{\delta_{\max}}{\Delta_{t_{\min}}}} = \Delta_{\max}. \tag{13}$$

6.2 Drift Compensation

Consider the update of the standard ATS algorithm (5). In the presence of delays, it becomes:

$$\begin{aligned} \hat{\alpha}_i(k) &= \hat{\alpha}_i(k-1) + [1 - \rho_v] \\ &\qquad \big[\alpha_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\big] \\ &= \hat{\alpha}_i(k-1) + [1 - \rho_v] \\ &\qquad \left[\frac{\alpha_j}{\alpha_i}\Delta_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\right]. \end{aligned} \tag{14}$$

We point out that the parameters $\hat{\alpha}_i,\ i = 1, \ldots, N$ computed according to (14) may diverge for some realizations of the delay. Intuitively, this can be explained by to the lack of symmetry in the pairwise update carried out by a pair of nodes i and j at each update k. To better understand this point, let us consider a simple WSN consisting only of two nodes. In particular, assume that node 1 updates at each odd k and node 2 updates at each even k.

Therefore, for an odd k the update becomes:

$$\begin{array}{l}\hat{\alpha}_1(k) = \hat{\alpha}_1(k-1)+[1-\rho_v]\left[\alpha_{12}(k)\hat{\alpha}_2(k-1)-\hat{\alpha}_1(k-1)\right] \\ \hat{\alpha}_2(k) = \hat{\alpha}_2(k-1)\end{array}, \tag{15}$$

and for an even k it becomes:

$$\begin{array}{l}\hat{\alpha}_1(k) = \hat{\alpha}_1(k-1) \\ \hat{\alpha}_2(k) = \hat{\alpha}_2(k-1)+[1-\rho_v]\left[\alpha_{21}(k)\hat{\alpha}_1(k-1)-\hat{\alpha}_2(k-1)\right]\end{array}. \tag{16}$$

Let assume that the following realization of the delay occurs:

$$\Delta_{ij}(k) = \Delta_{ji}(k) = \frac{1}{1-\epsilon}. \quad , \tag{17}$$

with an arbitrarily small scalar $\epsilon > 0$. Then, by using the coordinates transformation $\tilde{\alpha}_i = \alpha_i\,\hat{\alpha}_i$, $i \in \{1,2\}$ and assuming $\rho_v = 0.5$, the parameters evolution can be written as:

$$\begin{bmatrix}\tilde{\alpha}_1(2k+2)\\ \tilde{\alpha}_2(2k+2)\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & \frac{1}{1-\epsilon}\\ \frac{1}{1-\epsilon} & 1\end{bmatrix}\begin{bmatrix}\tilde{\alpha}_1(2k)\\ \tilde{\alpha}_2(2k)\end{bmatrix}. \tag{18}$$

The eigenvalues of the dynamical matrix in (18) are:

$$\lambda_1 = 0.5 - 0.5\frac{1}{1-\epsilon} \quad \text{and} \quad \lambda_2 = 0.5 + 0.5\frac{1}{1-\epsilon}, \tag{19}$$

the second of which makes the system unstable for any arbitrarily small scalar $\epsilon > 0$. This example shows that the ATS update rule proposed in [14] may experience problems if communication delays occur. Interestingly, this example is valuable also to understand where the problem lies. In this regard, consider the evolution of the sum and of the difference between the two parameters, namely $\tilde{\alpha}_s = \tilde{\alpha}_1 + \tilde{\alpha}_2$ and $\tilde{\alpha}_d = \tilde{\alpha}_1 - \tilde{\alpha}_2$. We have:

$$\begin{bmatrix}\tilde{\alpha}_d(k+2)\\ \tilde{\alpha}_s(k+2)\end{bmatrix} = \begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}\begin{bmatrix}\tilde{\alpha}_d(k)\\ \tilde{\alpha}_s(k)\end{bmatrix}. \tag{20}$$

From (20), it follows that, while the difference goes to zero, the sum keeps growing over time. This implies that the two variables reach the same value while growing over time. This clearly makes no sense in the context of the clock synchronization problem. Please note that with a different realization of the delays, e.g., $\Delta_{ij}(k) = \Delta_{ji}(k) = 1/(1+\epsilon)$, it is possible to have $\lambda_2 < 1$ which is even worse, as in that case both $\tilde{\alpha}_1$ and $\tilde{\alpha}_2$ converge to zero, meaning the two software clocks stop. We point out that, in absence of delays everything works as the sum of the terms is invariant. This implies that $\epsilon = 0$ and thus the eigenvalue $\lambda_2 = 1$. Thus the two $\tilde{\alpha}_i$ converge to the same bounded nonzero value. Based on this observation, the main idea of the RoATS is to recover this invariance property even in the presence of (random) bounded delays. In other words, the RoATS algorithm is built so as to ensure the fundamental properties at the basis of the consensus between two nodes, that are:

1) the two nodes variables must move one towards the other, which guarantees convergence;
2) the updates must be equal in magnitude and opposite in direction, which guarantees invariance.

To do so, the proposed update rule requires an additional communication round between each pair of nodes i and j in order to exchange the drift estimates $\alpha_{ij}(k)$ and $\alpha_{ji}(k)$. Indeed, by means of this information and of the bounds $\Delta_{\min}$ and $\Delta_{\max}$, each node can locally infer the following inclusions:

$$\frac{\alpha_{ij}(k)}{\Delta_{\max}} \le \frac{\alpha_j}{\alpha_i} \le \frac{\alpha_{ij}(k)}{\Delta_{\min}} \quad \text{and} \quad \frac{\Delta_{\min}}{\alpha_{ji}(k)} \le \frac{\alpha_j}{\alpha_i} \le \frac{\Delta_{\max}}{\alpha_{ji}(k)}, \tag{21}$$

from which it follows:

$$\underline{\eta}_{ij}(k) \le \frac{\alpha_j}{\alpha_i} \le \bar{\eta}_{ij}(k), \tag{22}$$

where:

$$\begin{aligned}\underline{\eta}_{ij}(k) &= \max\left\{\frac{\alpha_{ij}(k)}{\Delta_{\max}}, \frac{\Delta_{\min}}{\alpha_{ji}(k)}\right\},\\ \bar{\eta}_{ij}(k) &= \min\left\{\frac{\alpha_{ij}(k)}{\Delta_{\min}}, \frac{\Delta_{\max}}{\alpha_{ji}(k)}\right\}.\end{aligned} \tag{23}$$

At this point, each node can locally compute:

I) $\mu(k)$, which is the update direction of $\hat{\alpha}(k)$ and is defined as:

$$\begin{aligned}\mu(k) = &\frac{1}{2}\,\text{sign}\left[\underline{\eta}_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\right]\\ &+\frac{1}{2}\,\text{sign}\left[\bar{\eta}_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\right]\end{aligned}. \tag{24}$$

Note that, by construction $\mu(k) = 0$ in the case the two sign functions are opposite, which happens when a safe update direction cannot be decided.

II) $\Gamma_\alpha(k)$, which is the magnitude of the correction and is defined as:

$$\begin{aligned}\Gamma_\alpha(k) = \min\Big\{&\left|\underline{\eta}_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\right|,\\ &\left|\bar{\eta}_{ij}(k)\hat{\alpha}_j(k-1) - \hat{\alpha}_i(k-1)\right|,\\ &\left|\frac{1}{\underline{\eta}_{ij}(k)}\hat{\alpha}_i(k-1) - \hat{\alpha}_j(k-1)\right|,\\ &\left|\frac{1}{\bar{\eta}_{ij}(k)}\hat{\alpha}_i(k-1) - \hat{\alpha}_j(k-1)\right|\Big\}\end{aligned}. \tag{25}$$

Finally the update can be carried out by the pair of nodes i and j as follows:

$$\begin{aligned}\hat{\alpha}_i(k) &= \hat{\alpha}_i(k-1) + [1-\rho_v]\mu(k)\,\Gamma_\alpha(k)\\ \hat{\alpha}_j(k) &= \hat{\alpha}_j(k-1) - [1-\rho_v]\mu(k)\,\Gamma_\alpha(k)\end{aligned}, \tag{26}$$

where $\rho_v \in \left(1 - \dfrac{2\,\alpha_{\min}}{\alpha_{\max}+\alpha_{\min}},\, 1\right)$. Note that, with no lack of generality the parameters can be initialized as $\{\hat{\alpha}_i(0) = 1\}, \forall\, i = 1, ..., N$.

6.3 Compensation of the drift change on the offset

Let $t^i_{\alpha,k}$ be the time at which the i-th node performs the actual k update of the parameter $\hat{\alpha}_i$ on its local variable, which means:

$$\begin{aligned}\hat{\alpha}_i(t^{i,-}_{\alpha,k}) &= \hat{\alpha}_i(k-1)\\ \hat{\alpha}_i(t^{i,+}_{\alpha,k}) &= \hat{\alpha}_i(k).\end{aligned} \tag{27}$$

where $t^{i,-}_{\alpha,k}$ and $t^{i,+}_{\alpha,k}$ are the left-handed limit and right-handed limit of $t^i_{\alpha,k}$, respectively. The offset must be updated as follows:

$$\hat{o}_i(t^{i,+}_{\alpha,k}) = \hat{o}_i(t^{i,-}_{\alpha,k}) - \Delta\hat{\alpha}_i(k)\tau_i(t^i_{\alpha,k}), \tag{28}$$

where $\Delta\hat{\alpha}_i(k)$ is defined as:

$$\Delta\hat{\alpha}_i(k) = \hat{\alpha}_i(k) - \hat{\alpha}_i(k-1). \tag{29}$$

Note that, by substituting (27) and (28) in (2), the term (28) ensures $\hat{\tau}(t^{i,+}_{\alpha,k}) = \hat{\tau}(t^{i,-}_{\alpha,k})$, thus preventing discontinuities of the the software clock due to the drift updates.

Remark 2 *A direct consequence of* (28) *is that for any interval between* t *and* t' *such that no offset compensation occur, the software clock can be simply updated as:*

$$\hat{\tau}_i(t) = \hat{\tau}_i(t') + \int_{t'}^{t} \tilde{\alpha}_i(t).$$

7 Offset Compensation Protocol

The offset compensation is based on a single packet sent from node j and received by node i at time t_h. When a packet is received, node i stores the software clock $\hat{\tau}_i(t_h)$ at which the packet was received. The packet contains the tuple $(\mathrm{id}_j, \hat{\tau}_j)$, where $\hat{\tau}_j = \hat{\tau}_j(t_h - \delta_h)$ is the software clock time of node j at the time the packet was sent and δ_h is the transmission delay. We assume δ_h to be bounded from above, that is $\delta_h \leq \delta_{\max}$. Using only this information, the following offset compensation update is proposed:

$$\hat{o}_i(t^+_h) = \hat{o}_i(t^-_h) + [1-\rho_o][\hat{\tau}_j(\tilde{t}_h - \delta_h) - \hat{\tau}_i(t_h)], \tag{30}$$

under the following realistic and reasonable assumption:

- **Assumption 3**: The interval between the transmission of two consecutive offset compensation packets is upper bounded:

$$t_{h+1} - t_h \leq \Delta t^h_{\max}.$$

Remark 3 *In this work for the sake of simplicity we have assumed packet delays to be bounded as* $\delta \in [0,\ \delta_{max}]$. *The RoATS protocol and the related analysis can be adapted to cope with alternative characterizations of the delay. For example in the case of a delay distribution bounded between any two positive values, it would be enough to modify the algorithm by subtracting the average value of the interval in all the related updates. As it will be proven later, the smaller the bound of the uncertainty of the delay, the higher the synchronization accuracy.*

Remark 4 *By introducing further assumptions, such as slow variation of the channel latency, the RoATS could be complemented with an additional estimator of the delay's bounds. This would allow to further improve the clock synchronization accuracy by tuning the multiplicative factor (13) accordingly. Finally, we point out that stochastic modeling of the delay could be considered as well. For example, if a normal distribution is considered, a reasonable choice could be to set the maximum delay to 3 times the standard deviation as about 99.73% of the values lie within this interval. Nevertheless, a stochastic modeling of the delay would demand for a stochastic analysis of the protocol, which is beyond the scope of this paper.*

8 Algorithm Convergence Analysis

In this section, we characterize the convergence properties of the proposed RoATS algorithm. First, in Section 8.1 we show that the nodes' clock frequency converge to a bounded set in the presence of (random) bounded communication delays. Then, in Section 8.2 we show that the discrepancy between any pair of nodes' software clock remains bounded over time.

For the sake of the analysis, the coordinates transformation $\tilde{\alpha}_i = \alpha_i\,\hat{\alpha}_i$ will be used in the sequel. Accordingly:

I) The direction of the $\tilde{\alpha}(t_k)$ update becomes:

$$\mu(k) = \frac{1}{2}\,\text{sign}\left[\frac{1}{\alpha_i}\left(\frac{\alpha_i}{\alpha_j}\,\underline{\eta}_{ij}(k)\tilde{\alpha}_j(k-1) - \tilde{\alpha}_i(k-1)\right)\right] + \frac{1}{2}\,\text{sign}\left[\frac{1}{\alpha_i}\left(\frac{\alpha_i}{\alpha_j}\,\overline{\eta}_{ij}(k)\tilde{\alpha}_j(k-1) - \tilde{\alpha}_i(k-1)\right)\right], \tag{31}$$

II) The magnitude of the $\tilde{\alpha}(t_k)$ update becomes:

$$\begin{aligned}\Gamma_\alpha(k) = \min\Bigg\{ &\frac{1}{\alpha_i}\left|\frac{\alpha_i}{\alpha_j}\,\underline{\eta}_{ij}(k)\tilde{\alpha}_j(k-1)-\tilde{\alpha}_i(k-1)\right|,\\ &\frac{1}{\alpha_i}\left|\frac{\alpha_i}{\alpha_j}\,\overline{\eta}_{ij}(k)\tilde{\alpha}_j(k-1)-\tilde{\alpha}_i(k-1)\right|,\\ &\frac{1}{\alpha_j}\left|\frac{\alpha_j}{\alpha_i}\frac{1}{\underline{\eta}_{ij}(k)}\tilde{\alpha}_i(k-1)-\tilde{\alpha}_j(k-1)\right|,\\ &\frac{1}{\alpha_j}\left|\frac{\alpha_j}{\alpha_i}\frac{1}{\overline{\eta}_{ij}(k)}\tilde{\alpha}_i(k-1) - \tilde{\alpha}_j(k-1)\right|\Bigg\},\end{aligned} \tag{32}$$

III) The update for the pair of nodes (i, j) is:

$$\begin{aligned}\tilde{\alpha}_i(k) &= \tilde{\alpha}_i(k-1) + [1-\rho_v]\,\mu(k)\,\alpha_i\,\Gamma_\alpha(k)\\ \tilde{\alpha}_j(k) &= \tilde{\alpha}_j(k-1) - [1-\rho_v]\,\mu(k)\,\alpha_j\,\Gamma_\alpha(k)\end{aligned}. \tag{33}$$

Remark 5 *Note that, the nodes' clock frequency $\{\alpha_i\}, \forall\, i = 1, \ldots, N$ is unknown. Therefore, these equations cannot be implemented in practice and will be used only for the sake of the analysis.*

8.1 Clock Frequency Convergence

The following lemma shows that the difference between the parameters $\tilde{\alpha}_i$ and $\tilde{\alpha}_j$ does not increase each time that nodes i and j perform an update.

Lemma 1 *Consider a pair of nodes i and j performing the $(k+1)$-th update. Define the difference $\varepsilon_{ij}(k) = \tilde{\alpha}_i(k) - \tilde{\alpha}_j(k)$. If*

$$\rho_v \in \left(1 - \frac{2\,\alpha_{\min}}{\alpha_{\min} + \alpha_{\max}},\, 1\right),$$

the following holds:

i) The difference between the two updated parameters is not increasing, that is:

$$|\varepsilon_{ij}(k+1)| \le |\varepsilon_{ij}(k)|,$$

with $|\varepsilon_{ij}(k+1)| < |\varepsilon_{ij}(k)|$ if $\mu(k+1) \neq 0$ and $\varepsilon_{ij}(k+1) \neq 0$.

ii) The updated parameters belong to the convex combination of the previous ones, that is:

$$\tilde{\alpha}_i(k+1),\, \tilde{\alpha}_j(k+1) \in \text{conv}(\tilde{\alpha}_i(k),\, \tilde{\alpha}_j(k)).$$

Proof: The proof can be found in Appendix A.

Let us now state a sufficient condition to have an error-decreasing update.

Lemma 2 *A sufficient condition for a pair of nodes i and j performing the $(k+1)$-th synchronization step to have an update such that:*

$$|\varepsilon_{ij}(k+1)| < |\varepsilon_{ij}(k)|,$$

is:

$$\frac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} > \frac{\Delta_{\max}}{\Delta_{\min}} \quad \textit{or} \quad \frac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} < \frac{\Delta_{\min}}{\Delta_{\max}}. \tag{34}$$

Proof: The proof can be found in Appendix B.

The latter result provides a sufficient condition on the value of the ratio $\tilde{\alpha}_i(k)/\tilde{\alpha}_j(k)$ ensuring that the error $|\varepsilon_{ij}(k)|$ decreases, regardless of the particular realization of the delay. Note that, this condition is only sufficient and not necessary. In fact, as proved in Lemma 3, if $\tilde{\alpha}_i(k) \neq \tilde{\alpha}_j(k)$, then there always exists a nonsingular set of delay realizations which makes $|\varepsilon_{ij}(k)|$ decrease.

Lemma 3 *For any pair (i, j) for which $\tilde{\alpha}_i(t_k)/\tilde{\alpha}_j(t_k) > 1 + \epsilon$ with an arbitrary $\epsilon > 0$, there exists a scalar $\bar{\delta} > 0$ such that for $\forall\, \delta_1, \delta_2 \in [0, \bar{\delta}]$, the delay realization:*

$$\begin{aligned}\Delta_{ij}(k) &= \Delta_{\max} - \delta_1\\ \Delta_{ji}(k) &= \Delta_{\min} + \delta_2\end{aligned}, \tag{35}$$

ensures:

$$|\varepsilon_{ij}(k+1)| < |\varepsilon_{ij}(k)|.$$

Proof: The proof can be found in Appendix C.

The following lemma characterizes the equilibria subspace for the ideal case of instantaneous transmission, i.e., no transmission delay occurs. Notably, this turns out to be instrumental for the characterization of the convergence properties of the RoATS protocol.

Lemma 4 *Consider a WSN running the RoATS algorithm and assume $\Delta_{\min} = \Delta_{\max} = 1$. Assume that there exists a finite integer Δk such that for each update k the graph $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$ is connected, then $\text{span}(\mathbf{1})$ is the equilibria subspace. Furthermore, for any initial condition $\tilde{\alpha}(0) = [\tilde{\alpha}_1(0), \ldots, \tilde{\alpha}_N(0)]^T$, the unique equilibrium point is:*

$$\tilde{\alpha}^e = \kappa\,\mathbf{1}, \quad \kappa = \frac{\sum_{i=1}^N \frac{\tilde{\alpha}_i(0)}{\alpha_i}}{\sum_{i=1}^N \frac{1}{\alpha_i}} \tag{36}$$

with $\alpha = [\alpha_1, \ldots, \alpha_N]^T$ the local clock frequency of the nodes.

Proof: The proof can be found in Appendix D.

At this point, it is possible to prove that in the case of (random) bounded communication delays, the set where the nodes' software clock frequency will eventually converge is bounded and can be characterized as follows:

Theorem 2 *Consider a WSN running the RoATS algorithm and assume that there exists a finite interval Δk such that for any offset update k the graph $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$ is equal to $\mathcal{G}$*[1]*. If Assumptions 1, 2 hold true, then for each pair of nodes i and j the parameters $\tilde{\alpha}_1, ..., \tilde{\alpha}_N$ will eventually converge within a bounded set defined as:*

$$\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right)^{p_{ij}} \leq \frac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \leq \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right)^{p_{ij}}, \quad \forall i,\, j \in V, \tag{37}$$

with $p_{ij} < N$ the shortest path in terms of number of hops between the two nodes i and j in $\mathcal{G}$.

Proof: The proof can be found in Appendix E.

The following corollary shows that better convergence results can be achieved under opportune stochastic assumptions.

Corollary 1 *Consider a WSN running the RoATS algorithm and assume that there exists a finite interval Δk such that for any k the graph $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$ is equal to $\mathcal{G}$. Assume that for any positive scalar $\bar{\delta} > 0$ there exists a nonzero probability that the realization of the delay is* (35) *for some $\delta_1, \delta_2 \in [0,\, \bar{\delta}]$, then the expected value of all $\tilde{\alpha}_i(k), i = 1, .., N$ eventually converge, that is:*

$$\lim_{k \to \infty} E\left[\tilde{\alpha}(k)\right] = \tilde{\alpha}^e. \tag{38}$$

Proof: Lemma 3 ensures that if at time k the error ε_{ij} between two communicating nodes i and j is nonzero, then the probability that the two nodes will update is nonzero. By repeating the proof of Theorem 2 in view of this fact, the statement follows. $\square$

8.2 Software Clock Synchronization

The following theorem proves the robustness of the proposed RoATS algorithm against (random) bounded communication delays.

[1] Note that, as for the previous results, this requirement could be relaxed to the case of connectedness of $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$. This generalization is here omitted for the sake of clarity, as it would render the derivation of the bounds (37) significantly more involved.

Theorem 3 *Consider a WSN running the RoATS algorithm and assume that there exists a finite interval Δk such that for each update k the graph $\mathcal{G}(k, \Delta k) = (V, \{e_k, e_{k+1}, ..., e_{k+\Delta k}\})$ is connected. If Assumptions 1, 2 and 3 hold true, then the software clock difference $\hat{\tau}_j(t) - \hat{\tau}_i(t)$ is bounded for any $i, j \in V$ at any t. Moreover there exits a scalar $M_{i,j} > 0$ such that*

$$\lim_{t \to \infty} |\hat{\tau}_j(t) - \hat{\tau}_i(t)| < M_{i,j}, \quad \forall\, i, j \in V. \tag{39}$$

Proof: The proof can be found in Appendix F.

Remark 6 *The proof given in Appendix F is constructive and it allows for an explicit computation of $M_{i,j}$ if further assumptions on the nature of the communication scheme is taken, e.g., pre-defined communication schedule. Please refer to Appendix F for further details.*

9 Algorithm Validation

In this section, a validation of the RoATS clock synchronization algorithm is provided. In addition, a comparison against the standard ATS is described. First, the experimental testbed used for the validation is described in detail. Then, the results of the experimental validation are discussed. Finally, the results of the simulations algorithm are described.

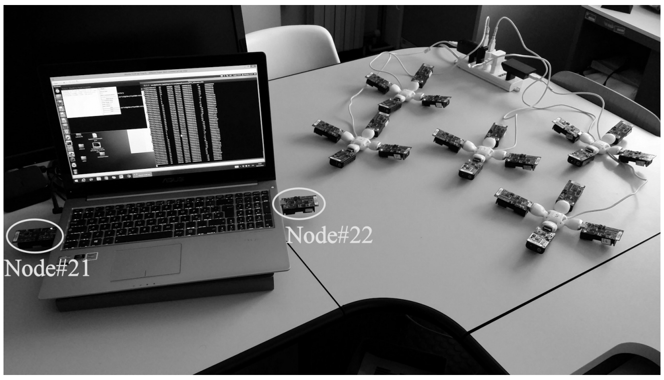

Fig. 1. Experimental testbed.

9.1 Experimental Testbed

The experimental testbed consisted of 22 Crossbow TelosB nodes and a PC. In particular, 20 nodes (indexed from 1 to 20) were used to implement the two synchronization algorithms, a node (indexed as 21) was plugged into the PC and used as ZigBee/USB protocol converter, and another node (indexed as 22) was used to trigger the 20 nodes to send the service messages containing their identification number ID_i, the software clock frequency $\hat{\alpha}_i$, the software clock offset $\hat{o}_i$ and the hardware clock τ_i value to the ZigBee/USB protocol converter (and in turn to the PC) at a given sampling time $T_{OS} = 2s$. The data collected by the PC was then used to evaluate the synchronization accuracy in terms of discrepancy among the software clocks $\hat{\tau}_i$ for all $i \in [1, 20]$. Fig. 1 shows the actual realization of the testbed.

The Crossbow TelosB mote (TPR2420) is an open source platform featuring an IEEE 802.15.4 radio with integrated antenna, a low-power micro-controller (MCU) with extended memory, USB programming capabilities and a sensor suite of light, temperature and humidity sensors. To address the clock synchronization problem the microprocessor, the clock and the radio play a fundamental role. In particular, the micro-controller is a TI-MSP430 featuring a Digitally Controlled Oscillator (DCO) running at 8MHz with a clock period $T_{DCO} = 0.125\mu s$ and 10kb of RAM. Furthermore, it also features an External Crystal Oscillator (ECO) running at 32768Hz, with clock period $T_{ECO} = 30.5\mu s$. Regarding the radio equipment, the TelosB features an IEEE 802.15.4/ZigBee compliant module equipped with a RF transceiver operating within the range $[2.4000 - 2.4835]$ GHz, compatible with ISM band, which allows a data rate of 250 kbps. An important feature of the radio chip CC2420 is the *MAC-layer* timestamp capability. This allows each node to read the local clock at the beginning of the transmission or reception of the Start Frame Delimiter (SFD) of a message, i.e the first bit. This mechanism strongly reduces the random delays introduced by the transmission and the readings of the synchronization messages. Indeed, this mechanism was used in [14] to support their major assumption, i.e., communication delays are negligible with respect to T_{clk}, i.e., the clock period. As the primary focus of our work is to deal with transmission delays, we decided to perform the timestamp without using this MAC layer time-stamping feature. In our implementation, τ is written on the packet by the micro-controller and then the message is passed to the radio to be sent. Indeed, this procedure, introduces random delays which are no longer negligible due to Carrier Sense Multiple Access (CSMA) policy implemented in the radio chip. We point out that, apart from providing a better scenario for the evaluation of the proposed RoATS algorithm, this choice was not cosmetic as there are several hardware platforms, such as the Crossbow's MicaZ mote, for which this represents the only option available to perform a timestamp. Thus, this reflects a realistic operating condition in several application contexts. The same reasoning applies for the choice of T_{clk}. In particular, since the *precision tag* "T32kHz" is not available for all the mote platform and considering that the transmission delay is equal to 17 ms (see Section 9.3 and [31]), the *precision tag* "TMilli" was adopted for the experimental evaluation. This gives the clock period $T_{clk} = 1/1024s \approx 0.98ms$.

TelosB motes have been programmed by using TinyOS, an open source operating system specifically designed for WSN [32]. A possible description of the RoATS implementation for a pair of nodes performing an update is given in Algorithm 1 and 2, where for the sake of clarity the time-dependence and the mechanism required to ensure the feasibility of the update, i.e., availability of all the required data and of the triggered node, are omitted. In particular, Algorithm 1 describes the set of operations carried out by the node triggering the update, while Algorithm 2 describes the set of operations carried out by the neighboring node triggered for the update. Note that, the packets related to the update of the offset are piggybacked within the packets of the first round of communication for the update of the clock frequency. The reason of this choice is twofold: on the one hand this leads to a simpler implementation; on the other hand this allows a fair comparison with the ATS protocol.

Algorithm 1 Triggering Node Protocol Description

```
1: procedure RoATS-Triggering-Node
2:     /* Node $j$ wakes up and selects a neighbor $i$ */
3:     $ID_i \leftarrow$ randSel($\mathcal{N}_j$)
4:     /* Node $j$ sends a packet $< ID_j, \hat{\alpha}_j, \hat{o}_j, \tau_j >$ */
5:     sendPkg($ID_i, < ID_j, \hat{\alpha}_j, \hat{o}_j, \tau_j >$)
6:     /* Node $j$ waits for a packet $< ID_i, \hat{\alpha}_j, \hat{o}_j, \tau_j, \hat{\alpha}_{ij} >$ */
7:     $< ID_i, \hat{\alpha}_j, \hat{o}_j, \tau_j, \hat{\alpha}_{ij} > \leftarrow$ recvPkg()
8:     /* Node $j$ apply the Offset Update */
9:     $\hat{o}_j = \hat{o}_j + (1 - \rho_o)(\hat{\tau}_i - \hat{\tau}_{ji})$
10:    /* Node $j$ computes Drift */
11:    $\hat{\alpha}_{ji} = (\tau_i - \tau_i^{\text{old}})/(\tau_{ji} - \tau_{ji}^{\text{old}})$
12:    /* Node $j$ sends a packet $< ID_j, \hat{\alpha}_{ji} >$ */
13:    sendPkg($ID_i, < ID_j, \hat{\alpha}_{ji} >$)
14:    /* Node $j$ computes Drift Compensation */
15:    $\hat{\alpha}_j = \hat{\alpha}_j + (1 - \rho_v)(\mu \Gamma_a)$
16:    /* Node $j$ compensate the Offset */
17:    $\hat{o}_j = \hat{o}_j - \Delta\hat{\alpha}_j \tau_{ji}$
18: end procedure
```

Algorithm 2 Triggered Node Protocol Description

```
1: procedure RoATS-Triggered-Node
2:     /* Node $i$ is triggered by a neighbor $j$ */
3:     $< ID_j, \hat{\alpha}_j, \hat{o}_j, \tau_j, \tau_{ij} > \leftarrow$ recvPkg()
4:     /* Node $i$ apply the Offset Update */
5:     $\hat{o}_i = \hat{o}_i + (1 - \rho_o)(\hat{\tau}_j - \hat{\tau}_{ij})$
6:     /* Node $i$ computes Drift */
7:     $\hat{\alpha}_{ij} = (\tau_j - \tau_j^{\text{old}})/(\tau_{ij} - \tau_{ij}^{\text{old}})$
8:     /* Node $i$ sends a packet $< ID_i, \hat{\alpha}_i, \hat{o}_i, \tau_i, \hat{\alpha}_{ij} >$ */
9:     sendPkg($ID_j, < ID_i, \hat{\alpha}_i, \hat{o}_i, \tau_i, \hat{\alpha}_{ij} >$)
10:    /* Node $i$ waits for a packet $< ID_j, \hat{\alpha}_{ji} >$ */
11:    $< ID_j, \hat{\alpha}_{ji}, \tau_{ij} > \leftarrow$ recvPkg()
12:    /* Node $i$ computes Drift Compensation */
13:    $\hat{\alpha}_i = \hat{\alpha}_i + (1 - \rho_v)(\mu \Gamma_a)$
14:    /* Node $j$ compensate the Offset */
15:    $\hat{o}_i = \hat{o}_i - \Delta\hat{\alpha}_i \tau_{ij}$
16: end procedure
```

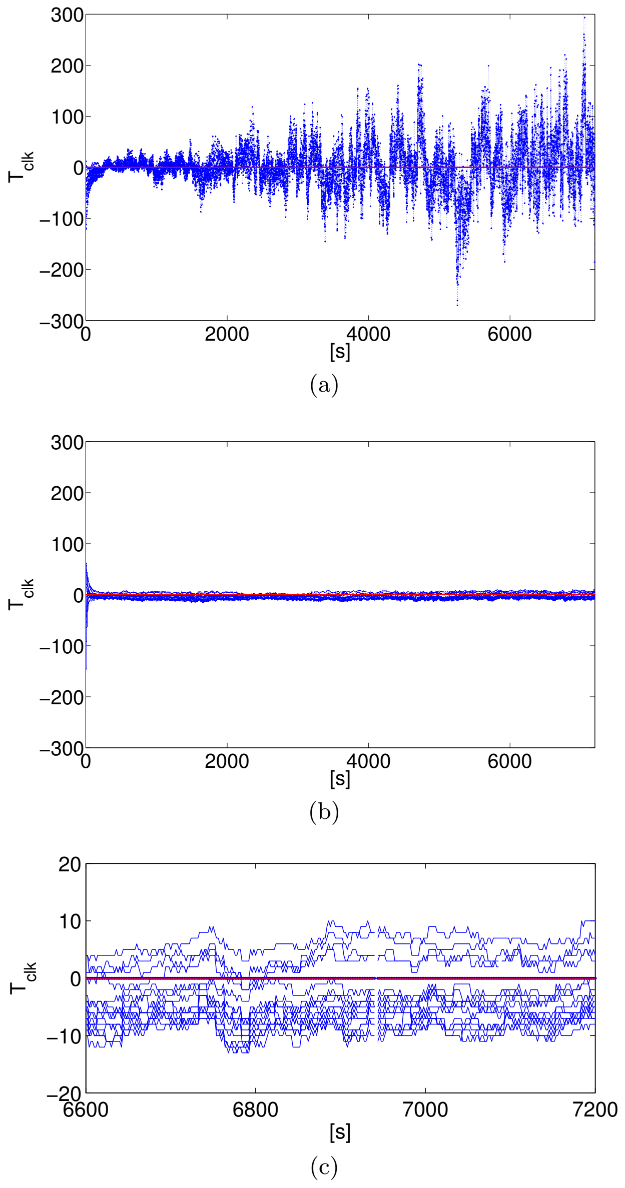

Fig. 2. Trend of the delay between nodes and node v_1 taken as reference is the case a) the ATS is used, and b) RoATS is used; c) zoom on the last 600 s in the case RoATS is used.

9.2 Experimental Results

Experiments were carried out to evaluate the effectiveness of the RoATS algorithm in a real-world scenario. For the experimental evaluation nodes were deployed in a region of approximately one square meter. The high spatial density of nodes was chosen to evaluate the effectiveness of the proposed algorithm in a saturated spectrum network. For each node i the neighborhood was predefined in order to achieve a lattice topology. For the experimental validation, the following parameters setting was used: $\Delta_{t_{\min}} = 10000\, T_{clk}$, $\Delta_{t_{\max}} = 10017\, T_{clk}$, $\rho_v = 0.9$ and $\rho_o = 0.9$. Fig. 2 shows the trend of the delay between node v_1 taken as a reference and the rest of the network in the case a) the ATS algorithm is used and b) the proposed RoATS algorithm is used. It can be noticed that, the discrepancy of the software clocks has an anomalous behavior in the case of the ATS algorithm due to the presence of communication delays, while it remains bounded in the case of the proposed RoATS. Fig. 2-c) shows that after the convergence to the bounded region is achieved, the maximum clock discrepancy among each pair of nodes is bounded by 20 T_{clk}. This corroborates the results of Theorem 3, where the difference of the software clocks between each pair of nodes was proven to remain bounded over time.

9.3 Simulation Results

Simulations were carried out in order to evaluate the scalability of the RoATS algorithm in a larger network setup. To this end, a wireless sensor network composed of 100 nodes deployed in a lattice topology with (random) bounded communication delays was considered. The same parameters setting as for the experimental evaluation was used. In addition, since the ECO runs at the frequency $f_{clk} = 32.768\text{kHz} \pm 20\text{ppm}$, the clock frequencies were chosen as $\alpha_i \in [0.999980, 1.000020] f_{clk}$, while the clock offsets were chosen as $\beta \in [0, 220] T_{clk}$. The communication delay was modeled as a uniform distribution $\mathcal{U}(0, \delta_{\max})$ where $\delta_{\max} = 17ms$ was experimentally measured.

Fig. 3 shows the trend of the delay between node v_1 taken as a reference and the rest of the network in the case a) the ATS algorithm is used and b) the proposed RoATS algorithm is used. As for the experiments with 20 nodes, also in the simulation with 100 nodes the discrepancy of the software clocks diverges in the case of the ATS algorithm due to the presence of communication delays, while it remains bounded in the case of the proposed RoATS. Fig. 3-c) shows that after the convergence to the bounded region is achieved, the maximum clock discrepancy among each pair of nodes is bounded by 20 T_{clk}. Indeed, this shows that the performance of the proposed RoATS algorithm scales well with the size of network. Clearly, this is a crucial point for the adoption of such a protocol in a real implementation.

Fig. 4 shows the trend of the $\tilde{\alpha}_i$ for all $i \in [1, 20]$ in the case a) the ATS algorithm is used and b) the proposed RoATS algorithm is used. We point out that this comparison can be carried out only in simulations as the actual local clock frequency α_i, $i \in V$ of the nodes required to compute $\tilde{\alpha}_i$, $i \in V$ is unknown in a real context. Note that, the software clock frequency of the nodes experiences a drift over time in the case of the ATS algorithm due to the presence of communication delays, while it remains within a bounded region in the case of the proposed RoATS. Indeed, this corroborates the results of Theorem 2, where the difference

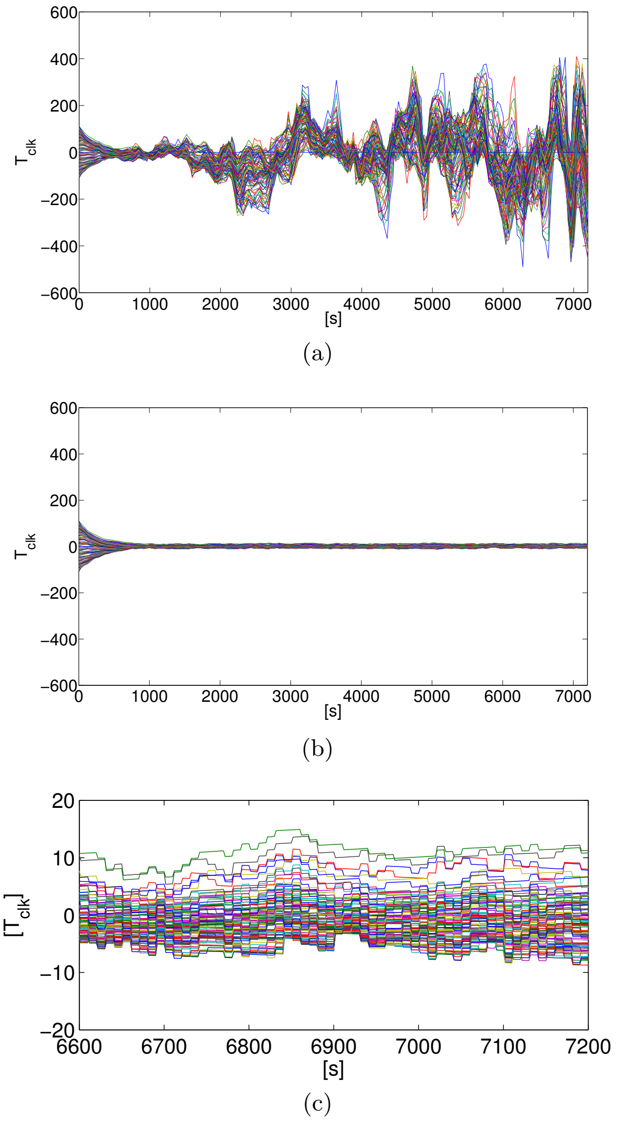

Fig. 3. RoATS Simulations: Trend of the delay between nodes and node v_1 taken as reference is the case a) the ATS is used, and b) RoATS is used; c) zoom on the last 600 s in the case RoATS is used.

between the software clocks frequency of each pair of nodes was proven to remain bounded over time. In particular, Fig. 4-c) shows that after the convergence to the bounded region is achieved, the maximum discrepancy of the software clock frequency among each pair of nodes is bounded by $1.6 \cdot 10^{-6} f_{clk}$.

10 Conclusion

In this work the clock synchronization problem for wireless sensor networks under the assumption of (random) bounded communication delays has been addressed. A novel robust algorithm consisting of two asynchronous consensus-based protocols for the synchronization of the

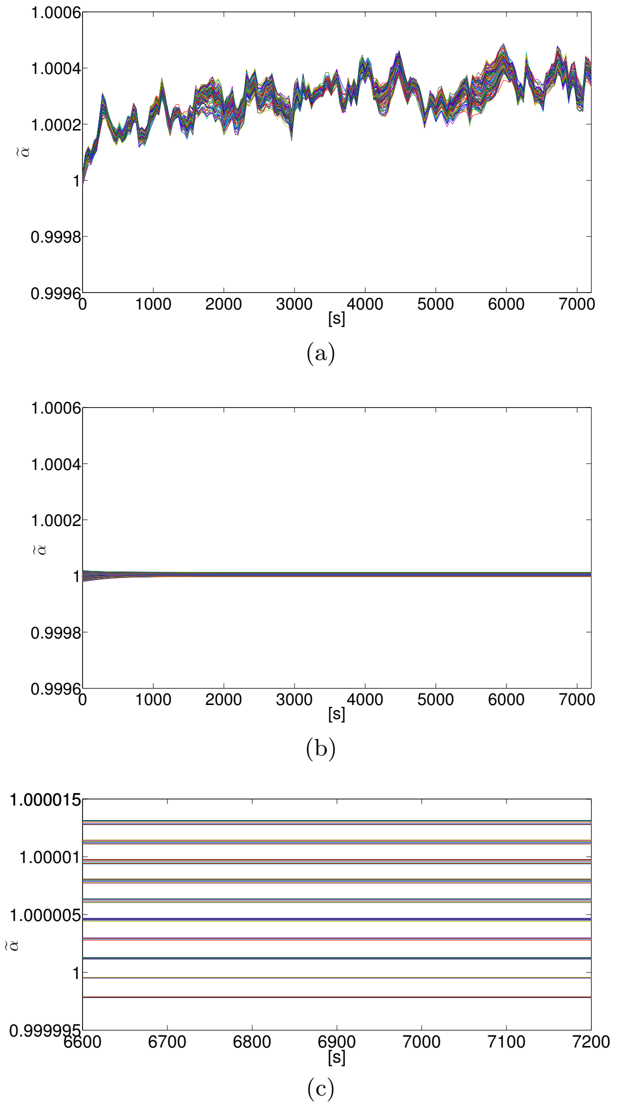

Fig. 4. ATS Simulations: Trend of the $\tilde{\alpha}_i$ for all $i \in [1, 20]$ in the case a) ATS is used and b) the proposed RoATS is used, c) zoom on the last 600 s in the case RoATS is used.

clocks' frequency and offset has been introduced. A theoretical analysis of the convergence and robustness properties of the proposed algorithm has been presented. Furthermore, simulations and experimental results have been presented which corroborate the theoretical analysis, evaluate the scalability of the proposed clock synchronization algorithm and show its effectiveness in a real-world scenario.

Appendix

A Proof of Lemma 1

Consider the error dynamics:

$$\varepsilon_{ij}(k+1) = \varepsilon_{ij}(k) + (1-\rho_v)(\alpha_i+\alpha_j)\mu(k+1)\,\Gamma_\alpha(k+1),$$

to prove the lemma we must show that the direction of the update $(1-\rho_v)(\alpha_i+\alpha_j)\mu(k+1)\,\Gamma_\alpha(k+1)$ is towards the negative gradient of the difference $\varepsilon_{ij}(k)$ and the magnitude is less than two times its absolute value. Note that, according to the control algorithm described in Section 6, two different scenarios might arise: either $\mu(k+1) = 0$ or $\mu(k+1) \neq 0$.

The first case $\mu(k+1) = 0$ is trivial as this implies $\varepsilon_{ij}(k+1) = \varepsilon_{ij}(k)$. For the second case $\mu(k+1) \neq 0$ consider the following equalities:

$$\begin{aligned} -\frac{\varepsilon_{ij}(k)}{\alpha_i} &= \frac{\alpha_j}{\alpha_i}\hat{\alpha}_j(k) - \hat{\alpha}_i(k) \\ +\frac{\varepsilon_{ij}(k)}{\alpha_j} &= \frac{\alpha_i}{\alpha_j}\hat{\alpha}_i(k) - \hat{\alpha}_j(k) \end{aligned}. \qquad \text{(A.1)}$$

From (22) it follows[2]:

$$\begin{aligned} -\frac{\varepsilon_{ij}(k)}{\alpha_i} &\in \operatorname{conv}\Big\{\underline{\eta}_{ij}(k+1)\hat{\alpha}_j(k) - \hat{\alpha}_i(k), \\ &\qquad \bar{\eta}_{ij}(k+1)\hat{\alpha}_j(k) - \hat{\alpha}_i(k)\Big\} \\ \frac{\varepsilon_{ij}(k)}{\alpha_i} &\in \operatorname{conv}\Big\{\frac{1}{\underline{\eta}_{ij}(k+1)}\hat{\alpha}_i(k) - \hat{\alpha}_j(k), \\ &\qquad \frac{1}{\bar{\eta}_{ij}(k+1)}\hat{\alpha}_i(k) - \hat{\alpha}_j(k)\Big\} \end{aligned}, \qquad \text{(A.2)}$$

Therefore, according to (24), the update is forced to move towards the negative gradient of the distance, that is $\mu(k+1) = -\operatorname{sign}(\varepsilon_{ij}(k))$.

Note that, this property is not sufficient to ensure the reduction of the distance $\varepsilon_{ij}(k)$ in absolute value, for instance the two estimates might even swap and get further away from each other. To this end, we need to ensure the following:

$$(1-\rho_v)\,(\alpha_i+\alpha_j)\,\Gamma_\alpha(k+1) < 2\,|\varepsilon_{ij}(k)|, \qquad \text{(A.3)}$$

[2] The use of the convex combination instead of an interval is required in (A.2) as it is not known a priori if $\underline{\eta}_{ij}(k+1)\hat{\alpha}_j(k) - \hat{\alpha}_i(k)$ is smaller than $\bar{\eta}_{ij}(k+1)\hat{\alpha}_j(k) - \hat{\alpha}_i(k)$ or vice versa.

where it should be noticed that the term $(1-\rho_v)\,(\alpha_i+\alpha_j)\,\Gamma_\alpha(k+1)$ is positive by construction. If $\mu(k+1) \neq 0$, from (25) and (A.2) it follows:

$$\Gamma_\alpha(k+1) \le \frac{|\varepsilon_{ij}(k)|}{\alpha_i} \ \text{ and } \ \Gamma_\alpha(k+1) \le \frac{|\varepsilon_{ij}(k)|}{\alpha_j}, \qquad \text{(A.4)}$$

which implies that the inequality (A.3) is satisfied if the following conditions hold true:

$$(1-\rho_v)\,\frac{(\alpha_i+\alpha_j)}{2\,\alpha_i} < 1 \ \text{ and } \ (1-\rho_v)\,\frac{(\alpha_i+\alpha_j)}{2\,\alpha_j} < 1. \qquad \text{(A.5)}$$

Since for $\alpha_i, \alpha_j \in [\alpha_{\min}, \alpha_{\max}]$

$$\min \frac{2\alpha_i}{\alpha_i+\alpha_j} = \frac{2\alpha_{\min}}{\alpha_{\min}+\alpha_{\max}},$$

condition (A.5) is verified if $\rho_v \in \left(1 - \frac{2\,\alpha_{\min}}{\alpha_{\min}+\alpha_{\max}},\, 1\right)$, which concludes the proof. □

B Proof of Lemma 2

By resorting to Lemma 1 the proof simply requires to show that condition (34) implies $\mu(k+1) \neq 0$.

As shown in (31), $\mu(k+1) \neq 0$ if and only if:

$$\begin{aligned} &\operatorname{sign}\left[\frac{1}{\alpha_i}\left(\frac{\alpha_i}{\alpha_j}\,\underline{\eta}_{ij}(k+1)\tilde{\alpha}_j(k) - \tilde{\alpha}_i(k)\right)\right] = \\ &\operatorname{sign}\left[\frac{1}{\alpha_i}\left(\frac{\alpha_i}{\alpha_j}\,\bar{\eta}_{ij}(k+1)\tilde{\alpha}_j(k) - \tilde{\alpha}_i(k)\right)\right] \end{aligned}. \qquad \text{(B.1)}$$

This can be re-written as:

$$\begin{cases} \dfrac{\alpha_i}{\alpha_j}\,\underline{\eta}_{ij}(k+1) > \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \\ \dfrac{\alpha_i}{\alpha_j}\,\bar{\eta}_{ij}(k+1) > \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \end{cases} OR \begin{cases} \dfrac{\alpha_i}{\alpha_j}\,\underline{\eta}_{ij}(k+1) < \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \\ \dfrac{\alpha_i}{\alpha_j}\,\bar{\eta}_{ij}(k+1) < \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \end{cases}. \qquad \text{(B.2)}$$

At this point by substituting (23) in (B.2), the previous set of inequalities becomes :

$$\begin{aligned} &\begin{cases} \max\left\{\dfrac{\Delta_{ij}(k+1)}{\Delta_{\max}}, \dfrac{\Delta_{\min}}{\Delta_{ji}(k+1)}\right\} > \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \\ \min\left\{\dfrac{\Delta_{ij}(k+1)}{\Delta_{\min}}, \dfrac{\Delta_{\max}}{\Delta_{ji}(k+1)}\right\} > \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \end{cases} \\ &\qquad OR \\ &\begin{cases} \max\left\{\dfrac{\Delta_{ij}(k+1)}{\Delta_{\max}}, \dfrac{\Delta_{\min}}{\Delta_{ji}(k+1)}\right\} < \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \\ \min\left\{\dfrac{\Delta_{ij}(k+1)}{\Delta_{\min}}, \dfrac{\Delta_{\max}}{\Delta_{ji}(k+1)}\right\} < \dfrac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \end{cases} \end{aligned}. \qquad \text{(B.3)}$$

Clearly (34) are sufficient conditions to satisfy (B.3). □

C Proof of Lemma 3

It is enough to prove that there exist a $\delta > 0$ such that the two sets of realization satisfy (B.3). Let us substitute (35) in (B.3). We obtain:

$$\begin{array}{l} \max\left\{\dfrac{\Delta_{\max}-\delta_1}{\Delta_{\max}}, \dfrac{\Delta_{\min}}{\Delta_{\min}+\delta_2}\right\} < 1+\epsilon \\ \min\left\{\dfrac{\Delta_{\max}-\delta_1}{\Delta_{\min}}, \dfrac{\Delta_{\max}}{\Delta_{\min}+\delta_2}\right\} < 1+\epsilon \end{array}.$$

Clearly for any $\epsilon > 0$ there exists a finite $\bar{\delta} > 0$ satisfying the statement of the lemma. □

D Proof of Lemma 4

In order to prove the lemma, let us note that in the nominal case without transmission delays we have $\Delta_{\min} = \Delta_{\max} = 1$ and thus the following holds:

$$\text{sign}\left[\frac{1}{\alpha_i}\left(\tilde{\alpha}_j(k) - \tilde{\alpha}_i(k)\right)\right] = \text{sign}\left[\frac{1}{\alpha_i}\left(\tilde{\alpha}_j(k) - \tilde{\alpha}_i(t_k)\right)\right].$$

According to (31), this implies that the only case where a pair of nodes i and j does not update their values, i.e., $\mu(k+1) = 0$, is when they are identical:

$$\tilde{\alpha}_j(k) = \tilde{\alpha}_i(k). \tag{D.1}$$

This implies that, if the graph is connected, the state of equilibrium $\tilde{\alpha}^{\text{e}} = [\alpha_1^{\text{e}}, ..., \alpha_N^{\text{e}}]^T$ must satisfy

$$\tilde{\alpha}_1^{\text{e}} = \tilde{\alpha}_2^{\text{e}} = \ldots = \tilde{\alpha}_{N-1}^{\text{e}} = \tilde{\alpha}_N^{\text{e}}, \tag{D.2}$$

which can be written as a linear system of $N-1$ equations in N unknowns. Since the update rule (26) is built to preserve the sum of the scalar parameters $\{\hat{\alpha}_i(k)\}, i = 1, \ldots, N$, then also the following equation will be satisfied at the equilibrium.

$$\sum_{i=1} \frac{\tilde{\alpha}_i^{\text{e}}}{\alpha_i} = \sum_{i=1} \frac{\tilde{\alpha}_i^{\text{e}}(0)}{\alpha_i} \tag{D.3}$$

Clearly, equations (D.2)-(D.3) admits as a unique solution (36), thus proving the lemma. □

E Proof of Theorem 2

Consider the Lyapunov Function $V(k)$:

$$V(k) = \left[\hat{\alpha}(k) - \hat{\alpha}^{\text{e}}\right]^T P\left[\hat{\alpha}(k) - \hat{\alpha}^{\text{e}}\right], \tag{E.1}$$

where $\hat{\alpha}^{\text{e}} = \left[\frac{\kappa}{\alpha_1}, ..., \frac{\kappa}{\alpha_N}\right]^T$ with κ as in (36) and $P = \text{diag}(\alpha_1, \ldots, \alpha_N)$.

If a pair of nodes i and j performs the $k+1$ update and if $\mu(k+1) \neq 0$, the difference of the Lyapunov Function $\Delta V(k) = V(k+1) - V(k)$ can be computed as follows:

$$\begin{aligned} \Delta V(k) = &\left[\hat{\alpha}(k+1) - \hat{\alpha}^{\text{e}}\right]^T P\left[\hat{\alpha}(k+1) - \hat{\alpha}^{\text{e}}\right] \\ &- \left[\hat{\alpha}(k) - \hat{\alpha}^{\text{e}}\right]^T P\left[\hat{\alpha}(k) - \hat{\alpha}^{\text{e}}\right] \end{aligned}. \tag{E.2}$$

According to the update law (33), only the parameters $\hat{\alpha}_i(k)$ and $\hat{\alpha}_j(k)$ are modified. Thus, (E.2) can be simplified as follows:

$$\begin{aligned} \Delta V(k) = &\,\alpha_i\left[\hat{\alpha}_i(k+1) - \frac{\kappa}{\alpha_i}\right]^2 - \alpha_i\left[\hat{\alpha}_i(k) - \frac{\kappa}{\alpha_i}\right]^2 \\ &+ \alpha_j\left[\hat{\alpha}_j(k+1) - \frac{\kappa}{\alpha_j}\right]^2 - \alpha_j\left[\hat{\alpha}_j(k) - \frac{\kappa}{\alpha_j}\right]^2 \end{aligned}. \tag{E.3}$$

By substituting the update (33) in (E.3), it follows that:

$$\begin{aligned} \Delta V(k) = &\,\alpha_i\left[\hat{\alpha}_i(k) + (1-\rho_v)\Big(\mu(k+1)\,\Gamma_\alpha(k+1)\Big) - \frac{\kappa}{\alpha_i}\right]^2 \\ &+ \alpha_j\left[\hat{\alpha}_j(k) - (1-\rho_v)\Big(\mu(k+1)\,\Gamma_\alpha(k+1)\Big) - \frac{\kappa}{\alpha_j}\right]^2 \\ &- \alpha_i\left[\hat{\alpha}_i(k) - \frac{\kappa}{\alpha_i}\right]^2 - \alpha_j\left[\hat{\alpha}_j(k) - \frac{\kappa}{\alpha_j}\right]^2 \end{aligned}. \tag{E.4}$$

This expression can be further simplified as follows:

$$\begin{aligned} \Delta V(k) = &\,2\,\alpha_i\left(\hat{\alpha}_i(k) - \frac{\kappa}{\alpha_i}\right)\left[(1-\rho_v)\,\mu(k+1)\,\Gamma_\alpha(k+1)\right] \\ &- 2\,\alpha_j\left(\hat{\alpha}_j(k) - \frac{\kappa}{\alpha_j}\right)\left[(1-\rho_v)\,\mu(k+1)\,\Gamma_\alpha(k+1)\right] \\ &+ (\alpha_i + \alpha_j)\left[(1-\rho_v)\Gamma_\alpha(k+1)\right]^2 \end{aligned} \tag{E.5}$$

that is:

$$\begin{aligned} \Delta V(k) = &\,2\,(1-\rho)\,\mu(k+1)\,\Gamma_\alpha(k+1)\left[\tilde{\alpha}_i(k) - \tilde{\alpha}_j(k)\right] \\ &+ (\alpha_i + \alpha_j)\left[(1-\rho_v)\Gamma_\alpha(k+1)\right]^2 \end{aligned} \tag{E.6}$$

At this point, it should be noticed that if $\mu(k+1) \neq 0$, combining (31) and (22) it follows that $\mu(k+1) = -\,\text{sign}(\tilde{\alpha}_i(k) - \tilde{\alpha}_j(k))$. Therefore the previous equation can be written as:

$$\Delta V(k) = -2\,(1-\rho)\,\Gamma_\alpha(k+1)\Big|\tilde{\alpha}_i(k)-\tilde{\alpha}_j(k)\Big| + (\alpha_i+\alpha_j)\,\Big[(1-\rho_v)\Gamma_\alpha(k+1)\Big]^2 \tag{E.7}$$

This implies that $\Delta V(k) < 0$ if the following holds:

$$(\alpha_i+\alpha_j)\,\Big[(1-\rho_v)\Gamma_\alpha(k+1)\Big]^2 < 2\,(1-\rho)\,\Gamma_\alpha(k+1)\Big|\tilde{\alpha}_i(k)-\tilde{\alpha}_j(k)\Big| \tag{E.8}$$

or equivalently by recalling that $\varepsilon_{ij}(k) = \tilde{\alpha}_i(k)-\tilde{\alpha}_j(k)$ and being $\Gamma_\alpha > 0$ by definition the following holds:

$$(\alpha_i+\alpha_j)\,(1-\rho_v)\Gamma_\alpha(k+1) < 2\,|\varepsilon_{ij}(k)| \tag{E.9}$$

As we already pointed out in Lemma 1, from (25) and (A.2) it follows that:

$$\Gamma_\alpha(k+1) \le \frac{|\varepsilon_{ij}(t_k)|}{\alpha_i} \quad \text{and} \quad \Gamma_\alpha(k+1) \le \frac{|\varepsilon_{ij}(t_k)|}{\alpha_j} \tag{E.10}$$

By summing the two inequalities we have that:

$$(\alpha_i+\alpha_j)\Gamma_\alpha(k+1) \le 2\,|\varepsilon_{ij}(t_k)| \tag{E.11}$$

which directly implies (E.9) by noticing that $\rho_v \in \Big(1-\frac{2\,\alpha_{\min}}{\alpha_{\min}+\alpha_{\max}},\,1\Big)$. Indeed, this proves that each time $\mu(k+1) \neq 0$ the Lyapunov function decreases, that is $\Delta V(k) < 0$.

At this point, by exploiting the result given in Lemma 2, we know that a sufficient condition for which $\mu(k+1) \neq 0$ is (34). This implies that, as time goes to infinity, since there exists an interval Δk, such that for each k the graph $\mathcal{G}(k,\Delta k)$ is equal to $\mathcal{G}$, the following holds

$$\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right) \le \frac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_j(k)} \le \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right), \quad \forall\,(i,\,j) \in E. \tag{E.12}$$

Therefore, for any pair $(i,j) \in V$ for which the shortest path is $p_{ij} = \{e_{is}, e_{sp}, \dots, e_{hq}, e_{qj}\}$ the following inequalities hold:

$$\begin{aligned}
\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right) &\le \frac{\tilde{\alpha}_i(k)}{\tilde{\alpha}_s(k)} \le \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right)\\
\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right) &\le \frac{\tilde{\alpha}_s(k)}{\tilde{\alpha}_p(k)} \le \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right)\\
&\vdots\\
\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right) &\le \frac{\tilde{\alpha}_h(k)}{\tilde{\alpha}_q(k)} \le \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right)\\
\left(\frac{\Delta_{\min}}{\Delta_{\max}}\right) &\le \frac{\tilde{\alpha}_q(k)}{\tilde{\alpha}_j(k)} \le \left(\frac{\Delta_{\max}}{\Delta_{\min}}\right)
\end{aligned} \tag{E.13}$$

Finally, by multiplying these inequalities we obtain (37), which concludes the proof. □

F Proof of Theorem 3

In order to prove the theorem, we will consider the update of the software clock at time instants t_h^-, i.e., the time just before an update of the software clocks parameters is carried out by any given pair of nodes i and j. For the sake of readability, the apex is omitted in the following, i.e., $t_h^- = t_h$.

Consider the h-th synchronization of the offset for node i that has just received a packet from node j. In view of Remark 2 and of (30), the software clock of the i-th node is:

$$\hat{\tau}_i(t_{h+1}) = \hat{\tau}_i(t_h) + (1-\rho)\Big[\hat{\tau}_j(t_h) - \hat{\tau}_i(t_h)\Big] + \tilde{\alpha}_i^h\Big[t_{h+1} - t_h\Big] - (1-\rho)\tilde{\alpha}_j(t_h-\delta_k)\delta_k$$

where the first and second terms together represents the synchronization update. The third term is the clock evolution due to the time and it is proportional to the average value $\tilde{\alpha}_i^h$ of the clock drift between t_h and t_{h+1} defined as $\tilde{\alpha}_i^h = \frac{1}{t_{h+1}-t_h}\int_{t_h}^{t_{h+1}} \tilde{\alpha}_i(\xi)\,\mathrm{d}\xi$. The last term represents the effect of the transmission delay. For all the other nodes the software clock evolves according to the time evolution as:

$$\hat{\tau}_w(t_{h+1}) = \hat{\tau}_w(t_h) + \tilde{\alpha}_w^h\Big[t_{h+1} - t_h\Big]$$

The overall update can be written in matrix form as:

$$\hat{\tau}(t_{h+1}) = A(h)\,\hat{\tau}(t_h) + \tilde{\alpha}^h\Big[t_{h+1} - t_h\Big] - u(h) \tag{F.1}$$

where:

- $A(h) = \big[I + (1-\rho_o)W(t_h)\big]$, with $W(t_h) = (e_i - e_j)^T(e_i - e_j)$ where e_i and e_j are the i-th and j-th column vectors of the canonical basis, respectively;
- $\tilde{\alpha}^h = [\tilde{\alpha}_1^h,\,\dots,\tilde{\alpha}_N^h]^T$;
- $u(h) = \Big[0,\dots\underbrace{(1-\rho)\tilde{\alpha}_j(t_h-\delta_h)\delta_h,}_{i-th}\dots,0\Big]^T$.

By exploiting Lemma 1, it follows that if $\alpha_i(0) = 1, i \in V$, then:

$$\tilde{\alpha}_i^h \in \Big[\alpha_{\min}, \alpha_{\max}\Big], \forall i \in V, h \ge 0. \tag{F.2}$$

This combined with $\delta_h \le \delta_{\max}$ allows us to bound u as follows:

$$\|u(h)\|_\infty \le (1-\rho_0)\alpha_{\max}\delta_{\max} \triangleq u_{\max}. \tag{F.3}$$

At this point let us introduce the following change of coordinates:

$$\begin{bmatrix} \hat{\tau}_1(t_h) \\ d(t_h) \end{bmatrix} = Q\hat{\tau}(t_h), \tag{F.4}$$

where:

$$Q = [e_1,\ e_2 - e_1, \dots,\ e_N - e_1]^T\,, \tag{F.5}$$

and where $e_i = [0,\ \dots,\ 1,\ \dots,\ 0]^T$ is the i-th vector of the canonical basis. By construction $Q^{-1} = Q$. Note that $d(t_h) = [\hat{\tau}_2(t_h) - \hat{\tau}_1(t_h), \dots, \hat{\tau}_N(t_h) - \hat{\tau}_1(t_h)]^T$ is the vector of the differences of each software clock with respect to the software clock of the first node. Furthermore, as a consequence of this change of coordinates the following holds:

$$\hat{A}(h) = Q\,A(h)\,Q = \left[\begin{array}{c|c} 1 & * \\ \hline \mathbf{0} & \tilde{A}(h) \end{array}\right], \tag{F.6}$$

which allows us to write the dynamics of the differences $d(t_h)$, independently of the value of $\hat{\tau}_1(t_h)$, as:

$$d(t_{h+1}) = \tilde{A}(h)d(t_h) + [I \quad -U] \begin{bmatrix} w(t_h) \\ u(h) \end{bmatrix}, \tag{F.7}$$

where:

$$w(t_h) = (t_{h+1} - t_h)\big[\tilde{\alpha}_2^h - \tilde{\alpha}_1^h, \dots, \tilde{\alpha}_N^h - \tilde{\alpha}_1^h\big]^T, \tag{F.8}$$

and $U = [e_2 - e_1,\ \dots,\ e_N - e_1]^T$.

Note that because of Assumption 3 we can also bound w as:

$$\|w(t_h)\|_\infty \le \big(\alpha_{\max} - \alpha_{\min}\big)\Delta t_{\max}^h \triangleq w_{\max}. \tag{F.9}$$

At this point, we can focus on the generic difference $\hat{\tau}_i - \hat{\tau}_1$ with $i \in V \setminus \{1\}$, and consider the following system:

$$\begin{aligned} d(t_{h+1}) &= \tilde{A}(h)d(t_h) + [I \quad -U] \begin{bmatrix} w(t_h) \\ u(t_h) \end{bmatrix}, \\ y_i(t_{h+1}) &= e_{i-1}^T\, d(t_{h+1}) \end{aligned} \tag{F.10}$$

with $y(t_{h+1}) = \hat{\tau}_i(t_{h+1}) - \hat{\tau}_1(t_{h+1})$. Since the unforced linear time-varying system $d(t_{h+1}) = \tilde{A}(t_h)d(t_h)$ is exponentially stable (as proven in [14, Theorem 6]), then according to [33] it is also BIBO stable. It follows that two positive scalars $l_{i,u}$ and $l_{i,w}$ exist such that:

$$\lim_{h\to\infty} |\hat{\tau}_i(t_h) - \hat{\tau}_1(t_h)| \le l_{i,u}u_{\max} + l_{i,w}w_{\max}, \tag{F.11}$$

where $l_{i,u}$ and $l_{i,w}$ are the ℓ_1 norm of the system (F.10) with respect to the input $u(t_h)$ and $w(t_h)$.

Note that, the following holds for any time instant $t \in (t_h,\ t_{h+1})$:

$$|\hat{\tau}_i(t) - \hat{\tau}_1(t_h)| \le |\hat{\tau}_i(t_h) - \hat{\tau}_1(t_h)| + (\alpha_{\max} - \alpha_{\min})\,t, \tag{F.12}$$

which concludes the proof. □

Remark 7 *Please note that in general the ℓ_1 norms cannot be easily computed a priori as they depend on the particular realization of the communication sequence. Nevertheless, there are cases, e.g., in the case of a deterministic periodic schedule, for which this can be done. Moreover it should be noticed that the bounds derived above establish an interesting relationship between the algorithm performance and the hardware specifications: $u_{\max}$ given in* (F.3) *is proportional to the maximal transmission delay $\delta_{\max}$, while $w_{\max}$ given in* (F.9) *is proportional to the maximum interval between two packet transmissions $\Delta t_{\max}^h$. Finally,* (F.8) *shows that at each time step $w(t_h)$ is proportional to the differences $\tilde{\alpha}_i^h - \tilde{\alpha}_1^h$ which decreases over time.*

References

[1] A. Gasparri, B. Krishnamachari, G. S. Sukhatme, A framework for multi-robot node coverage in sensor networks, Annals of Math and Artificial Intelligence (AMAI), Special Issue on Multi-Robot Coverage, Search, and Exploration 52 (2-4) (2008) 281–305.

[2] D. Ghataoura, J. Mitchell, G. Matich, Vigilant+: mission objective interest groups for wireless sensor network surveillance applications, IET Wireless Sensor Systems, 1 (4) (2011) 229–240. doi:10.1049/iet-wss.2011.0045.

[3] T. Harms, S. Sedigh, F. Bastianini, Structural health monitoring of bridges using wireless sensor networks, IEEE Instrumentation Measurement Magazine, 13 (6) (2010) 14–18. doi:10.1109/MIM.2010.5669608.

[4] T. Torfs, T. Sterken, S. Brebels, J. Santana, R. van den Hoven, V. Spiering, N. Bertsch, D. Trapani, D. Zonta, Low power wireless sensor network for building monitoring, IEEE Sensors Journal, 13 (3) (2013) 909–915. doi:10.1109/JSEN.2012.2218680.

[5] G. Hancke (Ed.), Industrial Wireless Sensor Networks: Applications, Protocols, and Standards, CRC Press, 2013.

[6] H. Ramamurthy, B. S. Prabhu, R. Gadh, A. Madni, Wireless industrial monitoring and control using a smart sensor platform, IEEE Sensors Journal, 7 (5) (2007) 611–618. doi:10.1109/JSEN.2007.894135.

[7] C. Fischer, H. Gellersen, Location and navigation support for emergency responders: A survey, IEEE Pervasive Computing, 9 (1) (2010) 38–47. doi:10.1109/MPRV.2009.91.

[8] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, S. Moulton, Sensor networks for emergency response: challenges and opportunities, IEEE Pervasive Computing, 3 (4) (2004) 16–23. doi:10.1109/MPRV.2004.18.

[9] R. Olfati-Saber, P. Jalalkamali, Coupled distributed estimation and control for mobile sensor networks, IEEE Transactions on Automatic Control, 57 (10) (2012) 2609–2614. doi:10.1109/TAC.2012.2190184.

[10] E. Xu, Z. Ding, S. Dasgupta, Target tracking and mobile sensor navigation in wireless sensor networks, IEEE Transactions on Mobile Computing, 12 (1) (2013) 177–186. doi:10.1109/TMC.2011.262.

[11] A. Gasparri, F. Pascucci, An interlaced extended information filter for self-localization in sensor networks, IEEE Transactions on Mobile Computing, 9 (2010) 1491–1504. doi:http://doi.ieeecomputersociety.org/10.1109/TMC.2010.122.

[12] H. Song, V. Shin, M. Jeon, Mobile node localization using fusion prediction-based interacting multiple model in cricket sensor network, IEEE Transactions on Industrial Electronics, 59 (11) (2012) 4349–4359. doi:10.1109/TIE.2011.2151821.

[13] F. Lamonaca, A. Gasparri, E. Garone, D. Grimaldi, Clock synchronization in wireless sensor network with selective convergence rate for event driven measurement applications, IEEE Transactions on Instrumentation and Measurement, 63 (9) (2014) 2279–2287. doi:10.1109/TIM.2014.2304867.

[14] L. Schenato, F. Fiorentin, Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks, Automatica 47 (9) (2011) 1878–1886. doi:10.1016/j.automatica.2011.06.012.

[15] A. Aziz, Y. Sekercioglu, P. Fitzpatrick, M. Ivanovich, A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks, IEEE Communications Surveys Tutorials, 15 (1) (2013) 121–144. doi:10.1109/SURV.2012.031612.00124.

[16] R. Williams, A. Gasparri, B. Krishnamachari, Route swarm: Wireless network optimization through mobility, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, 2014, pp. 3775–3781. doi:10.1109/IROS.2014.6943092.

[17] E. Garone, A. Gasparri, F. Lamonaca, Clock synchronization for wireless sensor network with communication delay, in: American Control Conference (ACC 2013), 2013, pp. 771–776.

[18] D. Mills, Internet time synchronization: the network time protocol, IEEE Transactions on Communications, 39 (10) (1991) 1482–1493. doi:10.1109/26.103043.

[19] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, Ad Hoc Networks 3 (3) (2005) 281–323.

[20] Y. C. Wu, Q. Chaudhari, E. Serpedin, Clock synchronization of wireless sensor networks, IEEE Signal Processing Magazine, 28 (1) (2011) 124–138. doi:10.1109/MSP.2010.938757.

[21] J. Elson, L. Girod, D. Estrin, Fine-grained network time synchronization using reference broadcasts, SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 147–163. doi:10.1145/844128.844143.

[22] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks, in: Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03, 2003, pp. 138–149.

[23] R. Carli, A. Chiuso, L. Schenato, S. Zampieri, Optimal synchronization for networks of noisy double integrators, IEEE Transactions on Automatic Control 56 (5) (2011) 1146–1152. doi:10.1109/TAC.2011.2107051.

[24] C. Liao, P. Barooah, Distributed clock skew and offset estimation from relative measurements in mobile networks with markovian switching topology, Automatica 49 (10) (2013) 3015–3022.

[25] Q. Chaudhari, E. Serpedin, K. Qaraqe, On maximum likelihood estimation of clock offset and skew in networks with exponential delays, IEEE Transactions on Signal Processing, 56 (4) (2008) 1685–1697. doi:10.1109/TSP.2007.910536.

[26] M. Leng, Y.-C. Wu, On clock synchronization algorithms for wireless sensor networks under unknown delay, IEEE Transactions on Vehicular Technology, 59 (1) (2010) 182–190. doi:10.1109/TVT.2009.2028147.

[27] J. S. Kim, J. Lee, E. Serpedin, K. Qaraqe, Robust clock synchronization in wireless sensor networks through noise density estimation, IEEE Transactions on Signal Processing 59 (7) (2011) 3035–3047. doi:10.1109/TSP.2011.2141660.

[28] R. Carli, A. Chiuso, L. Schenato, S. Zampieri, A pi consensus controller for networked clocks synchronization, IFAC World Congress on Automatic Control (IFAC 08).

[29] M. K. Maggs, S. G. O'Keefe, D. V. Thiel, Consensus clock synchronization for wireless sensor networks, IEEE Sensors Journal, 12 (6) (2012) 2269–2277. doi:10.1109/JSEN.2011.2182045.

[30] Crossbow technology, telosb, telosb mote platform, document part number: 6020-0094-01 rev b.

[31] K. Lui, Performance evaluation of zigbee network for embedded electricity meters, Master's thesis, KTC Electrical Engineering, Stockholm, Sweden (2009).

[32] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, Tinyos: An operating system for sensor networks, in: W. Weber, J. Rabaey, E. Aarts (Eds.), Ambient Intelligence, Springer Berlin Heidelberg, 2005, pp. 115–148.

[33] B. Anderson, Internal and external stability of linear time-varying systems, SIAM Journal on Control and Optimization 20 (3) (1982) 408–413. doi:10.1137/0320031.