A Distributed Consensus-based Clock Synchronization Protocol for Wireless Sensor Networks

Habib Aissaoua, Makhlouf Aliouat, Ahcène Bounceur, Reinhardt Euler

Noname manuscript No.
(will be inserted by the editor)

A Distributed Consensus-based Clock Synchronization Protocol for Wireless Sensor Networks

Habib Aissaoua[1,3] · **Makhlouf Aliouat**[2] · **Ahcène Bounceur**[3] · **Reinhardt Euler**[3]

Received: date / Accepted: date

Abstract The birth of computer networks and distributed systems has led to the appearance of the clock synchronization problem. This issue has gained increasing importance with the emergence of new resource constrained networks such as wireless sensor networks. In this paper, we propose a new distributed clock synchronization algorithm, referred to as Weighted Consensus Clock Synchronization (WCCS), whose objective is to achieve a consensus clock among network nodes. In this distributed approach and in contrast to centralized schemes, each node periodically exchanges the local clock reading with its immediate neighbor nodes. Then, each node employs these time informations to calculate its relative offset and skew with respect to its neighbor nodes using a weighted average consensus based technique. The effectiveness of WCCS is proved through both simulations and an experimental study on TelosB mote using TinyOS.

Keywords Consensus algorithm · Convergence · Clock drift · Wireless sensor networks · Distributed algorithms

1 Introduction

Since the advent of computer networks and distributed systems, clock synchronization has attracted intensive interest by researchers. This concern has recently been increased with the appearance of wireless sensor networks (WSNs)

Habib Aissaoua (✉)
E-mail: habib.aissaoua@gmail.com

[1]Department of Computer Science, University of Abderrahmane Mira, Bejaia, Algeria
[2]Department of Computer Science, University of Ferhat Abbas Setif 1, Algeria
[3]Lab-STICC CNRS laboratory, University of Western Brittany, Brest, France

which often have resource constrained devices. In fact, clock synchronization is crucial for a wide range of communication protocols and applications. Data fusion, MAC protocols and node sleep scheduling are some examples that require an accurate time to operate correctly. Indeed, the purpose of clock synchronization schemes is to ensure a common notion of time to the whole set of network nodes. However, these schemes are challenged by the restricted resources that typify nodes of such networks, the clock drifting caused by the imperfect hardware clocks used nowadays by network nodes, and also the uncertainty caused by message latency during communication between nodes. Therefore, clock synchronization protocols have to present a best compromise between synchronization accuracy and the available resources such as energy, memory storage and computation. In fact, synchronization schemes that have been designed for wired networks are not suitable for WSNs due to their particularity. Despite the scalability and robustness of network time protocol (NTP) [1] and the high accuracy of *Global Positioning System (GPS)* based synchronization techniques, these protocols are not appropriate for WSNs [2]. Mainly, the accuracy may be considered as a primary purpose when designing a synchronization scheme. However, in WSNs, other purposes such as *energy efficiency*, *scalability*, *robustness* and *convergence speed* also have to be taken into consideration.

Recently, the clock synchronization issue in WSNs has received a great deal of attention and evolved to a topic of considerable research interest. Consequently, numerous algorithms have appeared in the literature (e.g.,[3–8]) addressing the problem of sharing a common notion of time among network nodes. One particular area of research in distributed algorithms is the agreement or consensus issue which has also gained a lot of attention, particularly in the field of sensor networks, as a way to achieve a global solution in a totally decentralized fashion. The main idea behind consensus clock synchronization algorithms is to employ a linear iteration, in which each node updates its own local estimate of global clock value based on the time information received from its direct neighbors. As a result, all of the network nodes asymptotically converge to the same consensus clock. Based on such a distributed consensus algorithm, several approaches have been proposed in the literature to solve the clock synchronization problem in WSNs [6, 7, 9, 10].

In this paper, we propose a new distributed clock synchronization algorithm for WSNs called Weighted Consensus Clock Synchronization(WCCS). The key novelty of our contribution is to employ the elapsed time on arrival technique [11] during the synchronization process to deal with the low convergence speed that consensus techniques often suffer from. In addition, we adapt the exponential smoothing technique to our weighted average consensus algorithm in order to ensure an accurate clock skew synchronization. The idea of smoothing technique is to control the process that averages the data using a weighted combination between the previous and the newest data estimations. In fact, we have two reasons that motivate us to use this technique. First, because of possibly high clock drift variation caused by the change in ambient conditions, such as temperature for instance, the current estimate can

introduce more effect on the clock skew estimation than the prior ones [12]. Second, the authors of [13, 14] prove that allowing nodes to use their previous estimates at each iteration can enhance the robustness against frequent topological changes. Consequently, our proposed solution exploits such a technique by considering the current and previous skew estimations at each clock skew compensation process, in order to accurately estimate the relative clock skew, and to improve the robustness when the network topology changes dynamically. Also, our algorithm is fully distributed, where nodes communicate only with their direct neighbors in a very simple manner and without any prior knowledge of the network topology.

The rest of this paper is organized as follows. Section 2 describes some leading clock synchronization protocols found in the literature. In section 3, we will give some technical preliminaries followed by basic concepts and the problem definition, that are useful to understand our proposal. The design details of the proposed clock synchronization scheme are presented in Section 4. Simulation and experimental results followed by discussion and comparison of the proposed work with the most relevant work proposed in the literature are given in Section 5. Finally, Section 6 concludes the paper. Some of our proofs are presented in appendices A, B, C, and D.

2 Related Work

Clock synchronization in WSNs has been extensively studied, and thus many algorithms and protocols have been designed to address this issue. Generally, we can classify synchronization schemes into two categories: reference-based and distributed synchronization protocols. In the first category, the network nodes receive the time information from a reference node. Then, each node has to adjust its clock based on the clock information received from the reference node. In order to deal with the growth of the network size that typifies WSNs, a reference-based technique uses a hierarchical structure where the root node usually acts as a reference node. Indeed, this category of protocols remarkably enhances scalability and also has a better convergence speed compared to distributed synchronization protocols. However, it suffers from the frequent changes of topology and from node failures, and the synchronization error also grows exponentially with the size of the network [15]. In distributed synchronization schemes, however, each node performs synchronization with its direct neighbor nodes, and thus all node clocks will gradually converge to a common global time value. A distributed-based technique can thoroughly handle scalability without any prior knowledge of the network topology. Consequently, protocols using this technique are robust to node failures and network topology changes. Now, we briefly describe some of the most relevant work proposed in the literature. Authors in [3] have introduced a new clock synchronization scheme, called Reference Broadcast Synchronization (RBS), based on the assumption that neighboring nodes receive the same signal at nearly the same time. In RBS, the arrival time of a message broadcasted from a selected node

is recorded at every node, and then exchanged with neighboring nodes to estimate the relative clock offsets. The Timing-sync Protocol [4], called TPSN, uses a round trip synchronization technique in order to achieve a global time synchronization. To this end, after electing one node as the root node, a spanning tree will be built in which every node can synchronize itself with its parent node one level higher up in the tree. The Flooding Time Synchronization Protocol (FTSP) [5] is one of the most popular clock synchronization protocols in WSNs. In FTSP, the nodes collaborate to dynamically form an ad-hoc structure from which the root node floods the time information into the network. In order to achieve a high level of accuracy, it employs a customized MAC layer timestamp and uses the linear regression technique to estimate both the offset and the skew error with respect to the root node. In [11], authors have proposed a proactive time synchronization where nodes record the appearance time of any event of interest based on their local clocks. Hereafter, nodes that have perceived the event send the appearance time of that event while internodes carry out time transformation from the sender node to the corresponding local time, and thus the elapsed time from the event source til the arrival at the sink node will be computed. A broadcast gossip consensus algorithm has been proposed in [9] to achieve consensus among all network clocks. Using MAC layer timestamps, each node periodically broadcasts a synchronization packet that contains its local time, offset and drift estimates to its neighbors. Upon receipt, both the offset and clock drift compensation will be carried out using the gossip average scheme. In [16], the authors have proposed a maximum time synchronization protocol by which the fastest clock among neighbor nodes is elected to be a reference clock. As a result, all network clocks will asymptotically converge to the maximum clock value. The PulseSync protocol is proposed in [8] to deal with growth of the clock skew error relative to an enlargement of the network diameter. The basic idea of PulseSync is to rapidly disseminate the time information over the network in order to minimize the potential negative effects of clock drifting and the overall message latency. Authors in [17] have employed a clustering technique, based on the LEACH [18] protocol, to deal with the problem of high communication traffic and low convergence speed of distributed consensus time synchronization protocols. In [19], the authors have designed a scheme that employs the measured temperature to assist network nodes to automatically compensate the effect of the clock skew. In order to increase the interval between two synchronization periods without losing synchronization accuracy, the authors of [20] have conducted measurements during several months to prove the correlation of clock frequency and temperature. Then, the estimated correlation between the crystal frequency and the working temperature is used to directly remove the clock skew during a clock synchronization process. In [21], a new scheme based on clustering topology is proposed. Its basic idea is that during the synchronization process between the cluster-head and the reference node, all neighboring nodes can overhear the exchanged synchronization messages which are timestamped by the two nodes' local clocks. Therefore, the cluster members will take advantages of such synchronization traffic to adjust their

local clocks. In [22], a distributed receiver to receiver protocol is provided to cope with the need of a fixed reference which is the major drawback of the scheme introduced by RBS [3]. In [23], a fully-distributed synchronization algorithm has been proposed for WSNs under unknown exponential delays. The authors have exploited the round trip synchronization mechanism and the joint maximum likelihood estimator of clock skew, and the clock offset problem is cast into a linear programming problem.

3 Preliminaries and basic definitions

3.1 Network model

Generally, a WSN can be modeled as an undirected graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, consisting of a set of nodes $\mathcal{V} = \{1, ..., n\}$ and a set of edges set $\mathcal{E}(t) \subseteq \{\{i, j\} \mid i, j \in \mathcal{V}\}$ representing the available communication links which means that nodes i and j can reliably communicate with each other at time t. The symbol $\mathcal{N}_i(t) = \{j \in \mathcal{V} \mid \{i, j\} \in \mathcal{E}(t), i \neq j\}$ denotes the set of neighbors of node $i \in \mathcal{V}$, and $d_i(t)$ its degree, i.e., its number of neighbors where $d_i(t) \triangleq |\mathcal{N}_i(t)|$.

Assumption 1 *At any time t, the graph $\mathcal{G}(t)$ is connected (i.e., there is a path between any pair of distinct nodes in the network).*

Assuming $\mathcal{G}(t)$ to be connected, guarantees that each pair of its nodes can exchange messages within $\mathcal{G}(t)$. A weighted graph is a triple $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t), w(t))$, where $w : \mathcal{E} \longrightarrow \mathbb{R}_{>0}$ is a function assigning to each edge $\{i, j\} \in \mathcal{E}(t)$ a strictly positive real number w_{ij} called its weight.

3.2 Distributed average consensus

In WSNs, we can define a distributed consensus problem as the problem of making the individual nodes of a network to reach a consensus (or agreement) upon a common value of a quantity of interest allowing all the network nodes to cooperate in a coordinate fashion [24, 25]. A widely studied class of consensus issue is the average consensus problem in which each node holds a value or measurement in the aim to compute the average of all the values in the network using distributed linear iterations. Indeed, averaging algorithms employ a straightforward mechanism in a completely distributed manner. At every iteration, neighbor nodes exchange their local states (informations) and update their state based on the received data using a simple linear weighted average.

In order to describe the average consensus algorithm, let us assume that it progresses on a weighted graph $\mathcal{G}$. Further, we associate with each node $i \in \mathcal{V}$ some scalar value $x_i \in \mathbb{R}$ that defines its *state*. The goal of an average consensus algorithm is to compute the consensus value $\overline{m} = \frac{1}{n} \sum_{i=1}^{n} x_i$ in a distributed way. We suppose that the state of each node i at time slot t is denoted by $x_i(t)$, and the state of the network is a vector denoted as

$\mathbf{x}(t) = [x_1(t), ..., x_n(t)]^T$. Therefore, the iterative average consensus algorithm at iteration t can be written as [26, 27]:

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i} w_{ij}(t) x_j(t) \tag{1}$$

$$\textit{where} \qquad \sum_{j \in \mathcal{N}_i} w_{ij}(t) = 1 \qquad \textit{and} \qquad \begin{cases} w_{ij} > 0 & \textit{if } j \in \mathcal{N}_i \\ 0 & \textit{otherwise} \end{cases}$$

or equivalently as:

$$\mathbf{x}(t+1) = \mathcal{W}(t)\mathbf{x}(t) \tag{2}$$

where $W(t)$ gives the weight matrix at iteration t. The square matrix $W(t)$ is nonnegative if $w_{ij}(t) \geqslant 0$ for all i and j, and row-stochastic if it is nonnegative and the entries of each row sum up to 1. According to [28, 29], if $W(t)$ is row-stochastic and $\mathcal{G}$ connected, the iterative algorithm described by equation(2) asymptotically solves the average consensus problem, namely:

$$\lim_{t \to \infty} \mathbf{x}(t) = \overline{m}\mathbf{1}$$

where $\mathbf{1}$ is a column vector with all components equal to 1.

3.3 Clock model

Before delving into the details of our work, we first define the clock model that will be used in this paper. Each sensor node $i \in \mathcal{V}$ has its own local hardware clock consisting of an oscillator frequency that defines the rate f at which the clock progresses. As it is widely adopted, we associate with each node i a local clock $\tau_i(t)$ in order to implement an approximation of the hardware clock. An ideal clock i can be modelled as $\tau_i(t) = t$ for all t; thus, the first derivative of $\tau(t)$, that defines the rate f, has always to be equal to 1 (i.e., $\partial\tau(t)/\partial t = 1$). Unfortunately, the clock rate will deviate over time due to possible ambient condition changes such as temperature, and as usual we call this deviation the *drift rate* (ρ). Consequently, the local clock of a node i can be modelled as follows [30]:

$$\tau_i(t) = a_i t + b_i \tag{3}$$

where a_i is the clock *speed (rate)*, and b_i is the local clock *offset* at real time t. In fact, the parameters a_i and b_i cannot be calculated as the real time t is unavailable to network nodes. In order to cope with this problem, each node i has to measure its local time with respect to another node j. Thereby, assuming that $\tau_i(t)$ and $\tau_j(t)$ are the local clocks of nodes i and j, respectively, we can define the relative clock between them as:

$$\tau_i(t) = \hat{a}_{ij}\tau_j(t) + \hat{b}_{ij} \tag{4}$$

The parameters $\hat{a}_{ij}$ and $\hat{b}_{ij}$ represent the relative rate and the relative offset between the clocks of node i and j, respectively.

3.4 Problem definition

Sharing a common notion of time among network nodes is a crucial issue for most distributed systems. To achieve this purpose, each node has to exchange its local clock reading with its neighbor nodes or with a reference node. The process applied to serve the purpose of having a common time among network nodes is well known as *clock synchronization*. Unfortunately, due to the continuous clock drifting and the error introduced by the message delivery time, there is no algorithm that can exactly synchronize the node clocks [31, 32]; meaning that the goal of any synchronization scheme is to have the clocks quite closely synchronized. Therefore, if the offset between clocks is globally bounded by some specific constant δ, namely:

$$|\tau_i(t) - \tau_j(t)| \leq \delta \qquad \forall i, j \in \mathcal{V},\ i \neq j \tag{5}$$

we can declare that the network clocks are δ-synchronized.

3.5 Problem of interest

According to Inequality (5), the goal of clock synchronization algorithms is to ensure that, at any time, the node clocks may differ by at most δ. Therefore, our aim is to offer a distributed consensus-based synchronization scheme where each node strives to agree with its neighbors on the current time, thus allowing the whole set of network clocks to asymptotically converge to the consensus clock. Otherwise, the objective of our algorithm is to synchronize all network clocks with respect to a virtual consensus clock, given as:

$$\tau_v(t) = \alpha_v t + \beta_v \tag{6}$$

In fact, to maintain the continuity of the local clock $\tau_i(t)$, its value should not be modified directly. Consequently, our algorithm has to derive at each node $i \in \mathcal{V}$ a virtual compensated clock $\mathcal{C}_i(t)$ based on its local clock readings $\tau_i(t)$ and the received timing messages from its neighbors, namely:

$$\mathcal{C}_i(t) = \hat{\alpha}_i \tau_i(t) + \hat{\beta}_i \tag{7}$$

$$= \hat{\alpha}_i a_i t + \hat{\alpha}_i b_i + \hat{\beta}_i \tag{8}$$

As a result, from Equations (6)and(7), each node i has to find both values $\hat{\alpha}_i$ and $\hat{\beta}_i$ in order to converge its virtual compensated clock $\mathcal{C}_i(t)$ to the consensus clock $\tau_v(t)$ which means that:

$$\lim_{t \to \infty} \mathcal{C}_i(t) = \tau_v(t) \qquad \forall i \in \mathcal{V} \tag{9}$$

or equivalently:

$$\begin{cases} \lim\limits_{t \to \infty} a_i \hat{\alpha}_i(t) = \alpha_v \\ \lim\limits_{t \to \infty} \hat{\alpha}_i(t) b_i + \hat{\beta}_i(t) = \beta_v \end{cases} \tag{10}$$

Indeed, having the exact time information of other node clocks during the synchronization process is challenged by the uncertainty in communication latency. That is, each synchronization message will experience some delay before reaching the destination which may introduce some error. To cope with the error caused by message latency, we will adopt the elapsed time on arrival technique used in [11] that will be discussed in section 4.

4 The weighted consensus clock synchronization algorithm

In this section, we describe our proposed clock synchronization algorithm. The main idea of the algorithm is to employ a distributed iterative scheme based on linear consensus algorithms, where each node updates its own virtual compensated clock $\mathcal{C}_i(t)$ based on the time information received from its direct neighbors. Accordingly, all nodes will asymptotically converge to the same consensus clock $\tau_v(t)$. To this end, each node i tries to estimate both $\hat{\alpha}_i$ and $\hat{\beta}_i$ with respect to all its neighbor nodes $j \in \mathcal{N}_i$. Appropriately, our algorithm has to perform two basic stages for each synchronization round, which are skew error compensation and offset compensation.

4.1 Skew error compensation

The purpose of this stage is to force all network clocks to converge their clock rates to the virtual consensus clock rate. For that reason, each node i has to broadcast a synchronization packet that contains the timestamps of its own local clock $\tau_i(t)$ and the corresponding virtual compensated clock $\mathcal{C}_i(t)$ which are recorded at the application layer. Once node i begins to send the packet, after some delay in the MAC layer, it updates the field that contains the local clock $\tau_i(t)$ by subtracting it from the actual local clock reading. Now, the synchronization packet contains $\mathcal{C}_i(t)$ and the elapsed time since it has been recorded at the application layer, will be denoted by *age*. Just after the first byte has arrived, each receiver j records the arrival time using its local clock $\tau_j(t)$, that we denote by *arvT*, and updates the field that contains the *age* value by subtracting it from the arrival time (i.e., *arvT-age*). As a result, this field contains the value that expresses the local clock $\tau_j(t)$ of node j when the received packet has been created and timestamped at node i. Indeed, such a technique is useful when a node detects an event of interest and a neighbor node wants to know the time of this event. Thus, as long as the node clocks of the whole network have not achieved synchronization, each node may benefit from this technique in order to estimate its clock offset with respect to its neighbors. Therefore, each node i will receive from all its neighbor nodes $j \in \mathcal{N}_i$ a synchronization packet that contains the virtual compensated clock $\mathcal{C}_j(t)$ of a neighbor j and its local clock $\tau_i(t)$ that represents the time at which $\mathcal{C}_j(t)$ has been recorded. Using these two time informations, node i can calculate its relative clock rate $\hat{\alpha}_{ij}$ or clock skew error α_{ij} ($\alpha_{ij} = \hat{\alpha}_{ij} - 1$) with respect to node j.

Lemma 1 *Let* $offset_{ij} = \mathcal{C}_j(t) - \tau_i(t)$*, and let* t_1*,* $t_2 \in \mathbb{R}_{>0}$*, with* $t_2 > t_1$*, two time instances at which node* i *receives a synchronization packet from a neighbor node* j*. Then*

$$\alpha_{ij} = \frac{offset_{ij}^{t_2} - offset_{ij}^{t_1}}{\tau_i(t_2) - \tau_i(t_1)}$$

For a proof, see Appendix A

In our scheme, as opposed to most consensus clock synchronization schemes (e.g.,[7, 9, 10]) that require all nodes to communicate their estimated clock skew and clock offset to their neighbors, communicating these two time informations is not necessary and as a result, the size of the synchronization packets is reduced. Since packet size has a direct impact on the power consumption of transmission and reception , the consumed energy during communication will be reduced in our scheme. Hence, during a synchronization period, each node i computes its relative clock skew error α_{ij} using the time informations received from each neighbor j, and then calculates and updates its clock skew α_i with respect to all its neighbors using the following iterative consensus algorithm:

$$\alpha_i(k+1) = \sum_{j \in \mathcal{N}_i} w_{ij}(k)\alpha_{ij}(k) \tag{11}$$

where $\alpha_i(k+1)$ represents the updated state of the clock skew error of node i and w_{ij} denotes a weighting factor that has to be chosen at each synchronization round k. In our scheme, we adopt a time-varying weighting factor as the communication topology among nodes may dynamically be changed due to the unreliable nature of wireless links or to the limited range of radio communication. Namely, node j is a neighbor for i at time t only if they are within communication range. Therefore, if $j \in \mathcal{N}_i(t)$, then $w_{ij}(t) > 0$; otherwise, $w_{ij}(t) = 0$. Also, we employ a simplified strategy to compute the weights, in which nodes with high degree are assigned high weights. Thus, each node can distributively compute the weights as follows:

$$w_{ij}(t) = \begin{cases} \dfrac{d_j(t)}{\sum_{l \in \mathcal{N}_i} d_l(t)} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Clearly, the weighting factor in our strategy can easily be computed since it is only based on the exchange of local information; that is, a simple way to know the neighbors' degrees is that each synchronization packet has to carry the sender's degree information.

Theorem 1 *Using lemma 1 and according to the updated value of* α_i *on Equations* (11)*and*(12)*, and under Assumption 1, all clock rates converge to the same clock rate* α_v *meaning that:*

$$\lim_{k \to \infty} a_i \hat{\alpha}_i(k) = \alpha_v \qquad \forall i \in \mathcal{V}$$

For a proof, see Appendix B

Indeed, the change in ambient conditions may cause a high clock drift variation; thus, the current clock skew estimate can be more affected than the prior ones. To deal with such a problem, we perform the exponential smoothing technique on the previous and the newest clock skew estimations to accurately estimate the relative clock skew.

$$\alpha_i(k) = \lambda\alpha_i(k) + (1-\lambda)\alpha_i(k-1) \tag{13}$$

where λ denotes the weight factor that must be chosen between 0 and 1, and $\alpha_i(k), \alpha_i(k-1)$ define the newest and the previous clock skew error estimate, respectively, which are the outputs of Equation (11).

4.2 Offset compensation

During the skew error compensation stage, the local clock of each node is advanced and then, it is also necessary to compensate the possible offset errors. In order to minimize the communication overhead among network nodes, the offset compensation will be combined with the skew error compensation stage.

Lemma 2 *Let $offset_{ij}^{t} = \mathcal{C}_j(t) - \tau_i(t)$ be the offset at time t, and let $t, t_{old} \in \mathbb{R}_{>0}$, with $t > t_{old}$, be two time instances at which node i receives a synchronization packet from a neighbor node j. Then*

$$\mathcal{C}_j(t) = \hat{\alpha}_{ij}\tau_i(t) + offset_{ij}^{old} - (\hat{\alpha}_{ij} - 1)\tau_i(t_{old})$$

For a proof, see Appendix C

As a result, once node i receives all synchronization packets from neighbor nodes, that hold $\mathcal{C}_j(t)$ and $\tau_i(t)$, it can update its compensated clock $\mathcal{C}_i(t)$ as follows:

$$\mathcal{C}_i(t) = \hat{\alpha}_i\overline{\tau}_i(t) + offset_{i}^{old} - (\hat{\alpha}_i - 1)\overline{\tau}_i(t_{old}) \tag{14}$$

where $offset_i^{old}$ and $\overline{\tau}_i(t)$ can be computed using the same iterative algorithm as used to calculate the clock skew error α_i. Thus:

$$offset_i^{old} = \sum_{j\in\mathcal{N}_i} w_{ij}\, offset_{ij} = \sum_{j\in\mathcal{N}_i} w_{ij}(\mathcal{C}_j(t) - \tau_i(t)) \tag{15}$$

$$\overline{\tau}_i(t) = \sum_{j\in\mathcal{N}_i} w_{ij}\tau_i(t) \tag{16}$$

Theorem 2 *Under Assumption 1 and according to the update algorithms based on Equations (14),(15)and(16), all clock offsets converge to the same offset β_v meaning that:*

$$\lim_{k\to\infty} \hat{\alpha}_i(k)b_i + \hat{\beta}_i(k) = \beta_v$$

For a proof, see Appendix D

5 Validation of the Algorithm

5.1 Simulation results

In order to evaluate the WCCS algorithm, we show some simulation results based on the Castalia simulator for WSNs [33]. Indeed, the choice of such a simulator is not trivial since it is typified by many characteristics that allow the user to ideally test new protocols and/or algorithms before real experiments take place. For instance, Castalia features the most advanced wireless channel and provides an accurate radio model; it also considers *clock drift*, offers the possibility to measure CPU power consumption, and it can be easily extended in order to add new components [34]. In fact, Castalia randomly as-

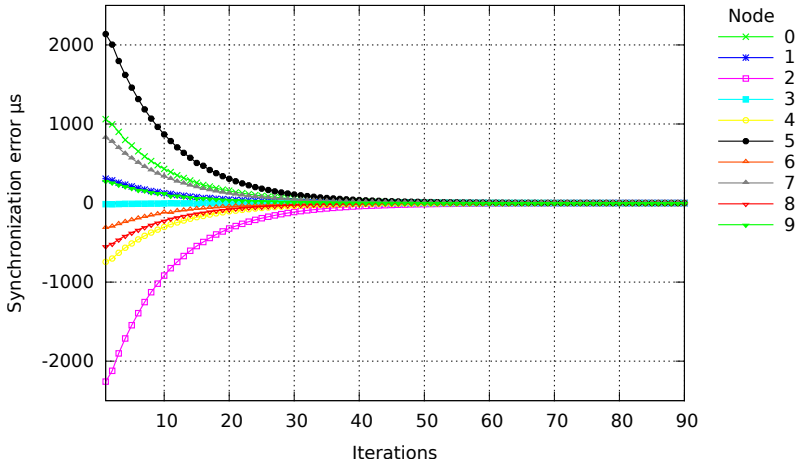

Fig. 1: Synchronization error with $\lambda = 0.1$

signs the clock drift to each node using a Gaussian distribution with zero mean and standard deviation of 30×10^{-6}, meaning that each clock can loose up to $\pm 30\mu s$ in one second. In order to accomplish the adopted elapsed time on arrival technique, we have introduced a MAC layer stamping into the Castalia simulator. To this end, the sender has to take a timestamp, and calculates the *age* in the radio layer once the *Start Frame Delimiter (SFD)* byte of the synchronization packet has been sent. Similarly and immediately after receiving the SFD byte, each receiver has to record its arrival time and to estimate its local clock value, that defines the time when the received packet has been created at the sender node, by subtracting the *age* value from the arrival time. We consider *offset* and *skew* errors, *scalability*, and *convergence* speed metrics to evaluate the performance of WCCS. The simulation parameters that we have used are shown in Table 1. First, we perform a simulation for one hour in which the synchronization algorithm starts after 45 minutes, and we

Table 1: Simulation parameters

Parameter name	**Value**
Number of nodes	$10, 25, 200$
Node deployment	Random
Radio transceiver	$CC2420$
Simulation time	1 hour, 2 hours
Simulation runs	50
Synchronization period	$10s$

use ten nodes, randomly deployed within an area of 20×20 meters, with a synchronization period equal to 10 seconds and a weight factor λ equal to 0.1. Upon receipt of the synchronization packet, each node i computes the current error between its virtual compensated clock and that of the sender node. In Figure 1, we have plotted the instantaneous mean synchronization error$(\frac{1}{\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} (\mathcal{C}_j(t) - \mathcal{C}_i(t)))$ at the end of every synchronization round. We can observe that the synchronization is achieved asymptotically and the error decreases progressively to zero after 40 rounds. Also, we can clearly see that the plotted data are smoothed by the exponential smoothing technique. In

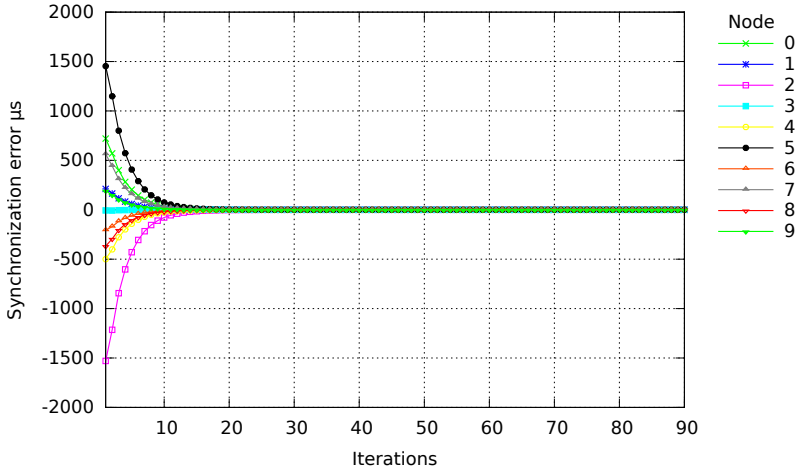

Fig. 2: Synchronization error with $\lambda = 0.3$

Figure 2, the weight factor λ is changed to 0.3, and as a result, the convergence speed is enhanced and the synchronization is achieved after just ten rounds. Indeed, the exponential smoothing technique efficiently reduces the impact of the possibly high clock drift variation, especially when ambient conditions are not suitable, by smoothing out the short-term fluctuations in frequency. However, this technique can be prejudicial to the convergence speed, especially, when we take a small value of λ as shown in Figure 1. To cope with such a

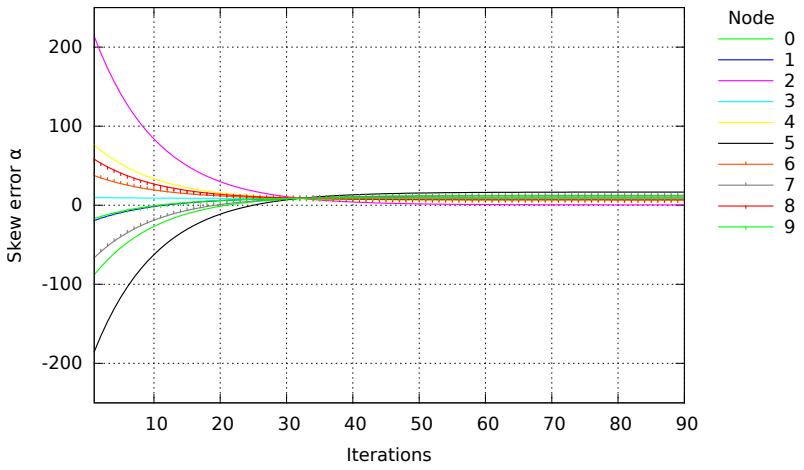

Fig. 3: Skew error with $\lambda = 0.1$

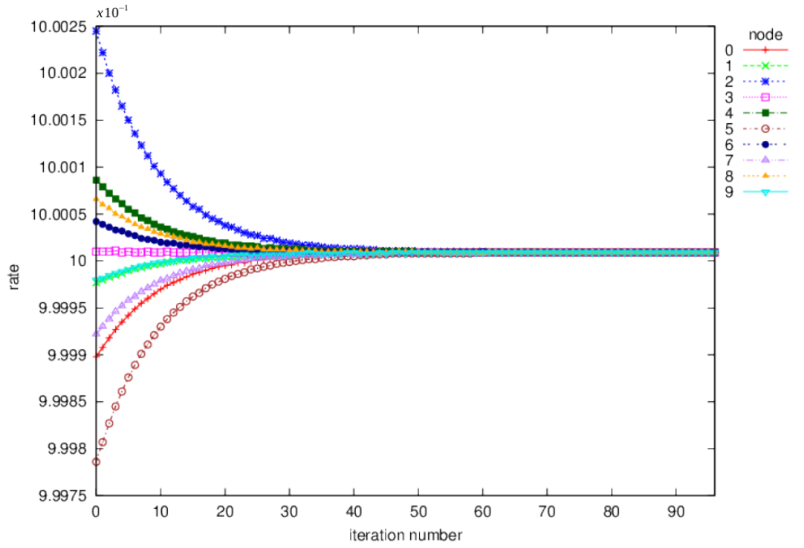

Fig. 4: Synchronization of clock rates with $\lambda = 0.1$

problem, neighboring nodes can estimate accurately their instantaneous offset error using the elapsed time on arrival technique while waiting the synchronization of all the network clocks to be achieved. In Figure 3, we can see that the skew error of each node decreases progressively and then remains stable after 40 rounds. The reason is that all node clocks have reached synchronization, as we have seen in Figure 1, and thus each node will maintain an error level

with respect to all node clocks. As claimed in Theorem 1, the Figure 4 shows the convergence of all clock rates to a unique one. Until now, all simulations

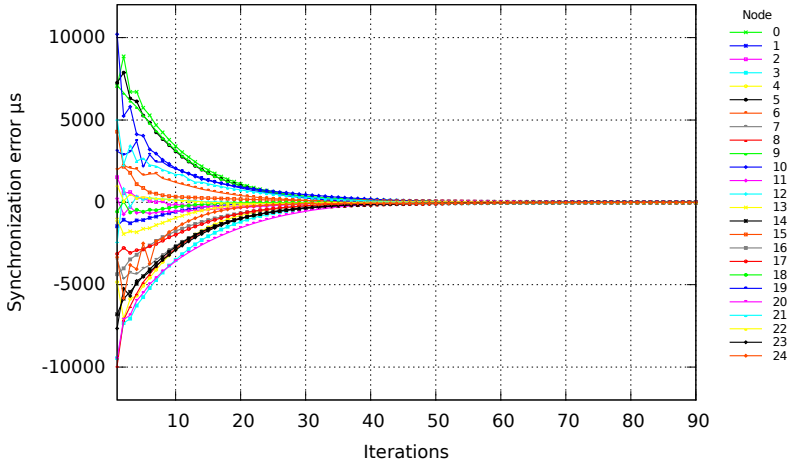

Fig. 5: Synchronization error with $\lambda = 0.1$

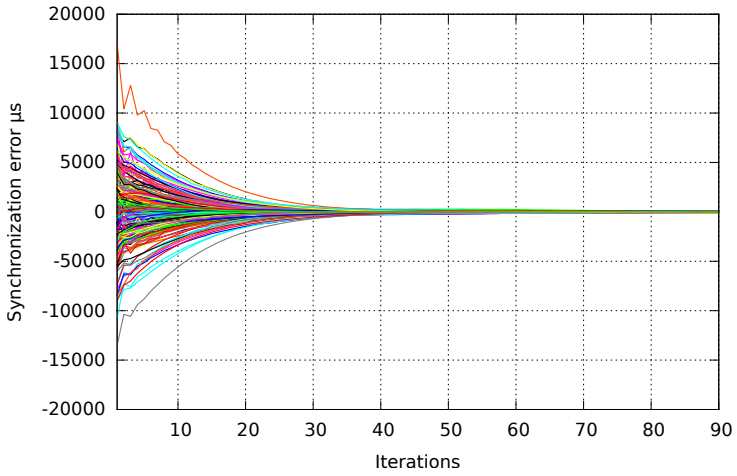

Fig. 6: Synchronization error with $\lambda = 0.1$

have been carried out for the case that all communicating nodes are one-hop away from each other. In order to perform multi-hop synchronization, we have run a simulation with 25 nodes randomly deployed in an area of 120×120 meters. From Figure 5 we can see that, during 40 rounds, the clock error of each

node gradually decreases and converges to zero. As a result, the convergence speed is not affected and does not decrease with an increase of the number of nodes. To further prove the scalability of our scheme, we increase the number of nodes to 200 and we randomly place them in an area of 350×350 meters. According to Figure 6, we can observe that the convergence speed is not really affected by the network size which means that our scheme is scalable.

5.2 Experimental results

In order to verify that our theoretical and simulation results can also be applied to real sensor networks, we present experimental results for our algorithm WCCS. To that end, we have implemented our proposed protocol on the TelosB platform, a research oriented mote, developed by UC Berkeley. Each TelosB mote is powered by a Texas Instruments MSP430 micro-controller, an internal digitally controlled oscillator running at 1 Mhz, an external oscillator running at 32kHz, and a USB port for programming and communication. In addition, TelosB motes are equipped with a Chipcon CC2420 radio module compatible with IEEE 802.15.4 (ZigBee) standard that operates at 2.4 GHz band. The Radio chip CC2420 offers a data rate of 250 kbps, and it is characterized by the capability to timestamp the sent and received packets at the MAC layer. The implementation of our protocol on the TelosB platform is done

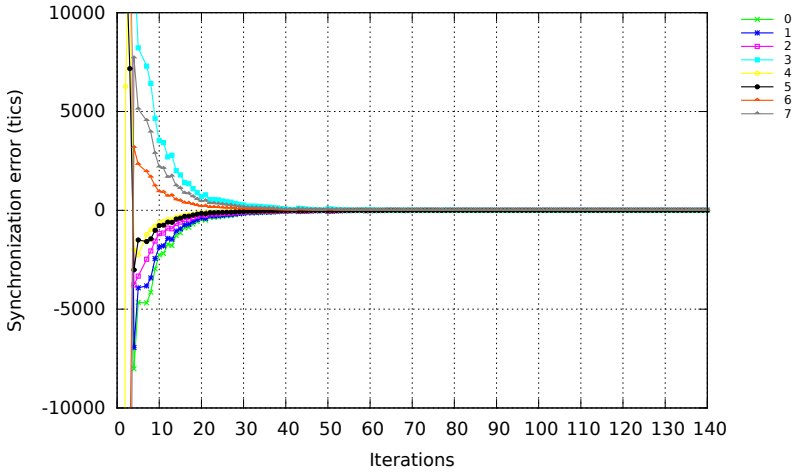

Fig. 7: Synchronization error with $\lambda = 0.1$

using TinyOS 2.1.2, an open source operating system designed for WSNs. As the 32kHz external clock runs even in idle mode, we have attached all node times to that clock in order to obtain a clock source. However, the resolution of 32kHz crystal is $30.5\mu s$ (one clock tick), which means that the maximum clock

accuracy will be limited to 30.5μs. Indeed, TinyOS allows to apply the elapsed

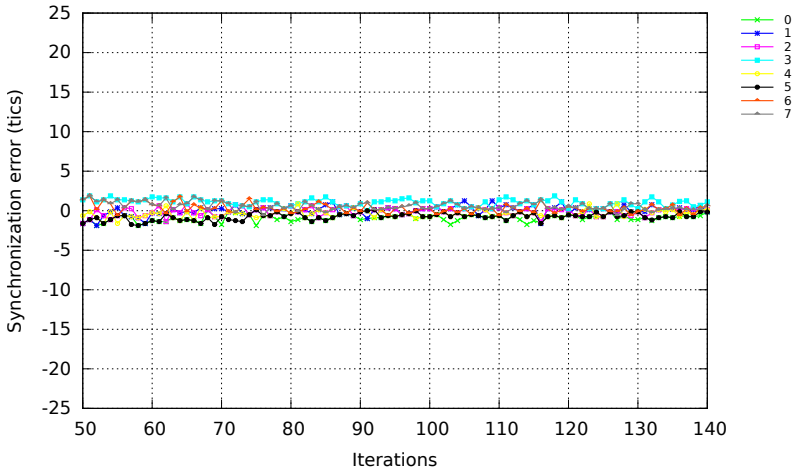

Fig. 8: Synchronization error with $\lambda = 0.1$

time on arrival technique by using *TimeSyncAMSend* and *TimeSyncPacket* interfaces. The send command provided by the *TimeSyncAMSend* interface has a parameter *event_time* that holds the timestamps of an event expressed on the local clock of the caller. Each receiver uses the *eventTime* command provided by the *TimeSyncPacket* interface in order to get the time of this event expressed in its own local clock. Our testbed is composed of 8 nodes deployed within a single broadcast domain with one node plugged into the PC in order to collect the experimental results. The weight factor λ is chosen to be equal to 0.1 and the synchronization period is 10 seconds. Every 9 seconds, the node plugged into the PC will broadcast a request packet asking all nodes to report the arrival time of this packet using their virtual compensated clocks. Upon receipt of their responses, the instantaneous clock errors between nodes will be computed by subtracting the average of all virtual compensated clocks from that of each node i (error= $\mathcal{C}_i(t) - \frac{1}{n}\sum_{j=1}^{n}\mathcal{C}_j(t)$). As we have articulated in Theorems 1 and 2, we show in this experiment the convergence of our algorithm and the steady state of all virtual compensated clocks. Figure 7 shows the convergence of WCCS, meaning that consensus is asymptotically achieved and the synchronization error decreases progressively to zero. Also, we can see the influence of the exponential smoothing technique: the potential clock drift variation caused by oscillator's instability, change in supply voltage, temperature, etc. is reduced, and thus the plotted data are smoothed. Figure 8 plots the instantaneous clock errors between all nodes starting from round 50. We can clearly see that the synchronization error over all nodes remains bounded, which means that all virtual compensated clocks have reached a steady state.

We also observe, that the error is bounded within ± 2 clock ticks, i.e., $\pm 61\mu s$, which implies the stability of our proposed synchronization scheme.

5.3 Discutions and comparison between WCCS and FTSP

In fact, the design of clock synchronization protocols for WSNs is challenging because it is typified by various network constraints. As a result, it is practically infeasible to design a totally efficient scheme. For instance, to achieve a high accuracy level we have to increase the number of timing messages when performing synchronization [3]. Unfortunately, this high level of precision is obtained to the detriment of energy efficiency and communication overhead as the number of synchronization messages is increased. Another important challenge is scalability, meaning that the synchronization shceme has to operate in large networks without losing its performance. Generally, it can be handled by adopting a hierarchical topology based technique. However, such network organization is prejudicial to the robustness when the topology changes frequently, and to the synchronization accuracy for the faraway nodes in the hierarchy structure. For that raison, each clock synchronization scheme has its distinctive advantages and drawbacks. Consequently, researchers have to investigate possible trade-offs between different approaches in order to achieve a reasonable synchronization scheme. Now, we compare our approach with FTSP [5], a standard benchmark clock synchronization protocol, to show its effectiveness and weaknesses. The FTSP is a popular reference-based synchronization scheme that also employs the MAC layer timestamp. The experimental results reported by the authors of FTSP show that the achieved average error is $3\mu s$, whereas the experimental results, which have been shown in Figure 8, indicate an average synchronization error of $61\mu s$ achieved by WCCS. Therfore, we can clearly see that the accuracy of FTSP is better than our scheme. Nevertheless, both of FTSP and our scheme achieve a high level of accuracy as it is in the order of micro-seconds. In addition, both of FTSP and WCCS have been designed with the assumption that the propagation delays among sensor nodes is negligible. As a result, they are unsuitable for high latency networks (e.g., underwater sensor networks UWSNs) due to the slow propagation speed that characterizes such networks. Besides, FTSP uses the method of least square linear regression to estimate both the offset and skew error with respect to the reference clock. In order to achieve a better clock drift estimation with the linear regression method, the number of the collected timing data has to be increased. Clearly, such a method is not suitable for networks with limited memory, such as wireless sensor networks. Also, the least square regression requires significant amount of computation time [35]. In addition, least-square regression exhibits a poor performance since sensor nodes which are far away from the reference node collect the estimated reference time with large deviations due to the waiting times at each sensor node [36]. As a result, the slopes of their least-square line exhibit large errors when compared to that of nearby nodes to the reference. Thus, such a problem affects negatively the scalability of

FTSP. On the contrary, the storage overhead in our algorithm is related to the number of neighbor nodes as each node has to store only the last received clock value from each neighbor. Since the size of neighboring nodes is usually small even for large networks, our algorithm requires little memory. Also, compared to the least square regression, our algorithm adopts a simple calculation methode to compute both the offset and skew error, and thus, the time and space complexities are quite small. The dynamic topology changes including node mobility, the death of a root or a non-leaf node, link failure or link addition when the network upgrades, have been treated by the FTSP authors by using periodic flooding of timing messages, and implicit dynamic topology update. On the other hand, a new root-election or parent-discovery procedure should be done once a change in the network occurs. The drawback of such a solution is that it requires large communication overhead especially when the network topology changes frequently. Also, [8] confirms that the synchronization error is accumulated quickly as the size of the network grows, and the delay at each node caused by the waiting time at the MAC layer may markedly decrease its accuracy and scalability. In contrast, our algorithm is fully decentralized, and thus, robust against node failure and dynamic topology changes, and it operates without any predetermined network topology and any reference clock. However, as WCCS belongs to the family of consensus-based approaches which are characterized by a low convergence speed, the convergence speed of FTSP is better than that of WCCS. To deal with the low convergence speed, we have involved the elapsed time on arrival technique [11] during the synchronization process. Thus, each node may benefit from this technique in order to estimate its clock offset with respect to its neighbors as long as the node clocks of the whole network have not achieved synchronization.

6 Conclusion

In this paper, we have investigated clock synchronization in WSNs and presented a new fully distributed synchronization algorithm called weighted consensus clock synchronization (WCCS). In WCCS, the nodes periodically broadcast a synchronization packet to their neighbors using the elapsed time on arrival technique. Using a weighted iterative consensus algorithm, each node computes both skew and offset errors and applies the exponential smoothing technique to the resulted clock skew in order to smooth out the short-term fluctuations in frequency caused by the variations in ambient conditions. We have proved by theoretical analysis that by applying our algorithm, all network clocks converge to a common clock value within a finite time. Using the Castalia simulator, an appropriate simulator for WSNs, we could emphasize our theoretical results through extensive simulation results. Furthermore, experiments on the TelosB platform have been carried out to corroborate the simulation and theoretical results and show the effectiveness of WCCS in realistic scenarios. Our future work includes a comparative study between our solution and the most relevant work proposed in the literature.

Appendix A Proof of Lemma 1 (see page 9)

Proof Once node i receives a synchronization packet at time t_1, it tries to synchronize itself with the received virtual compensated clock $\mathcal{C}_j(t)$. The relative clock rate is the ratio of the virtual compensated clock frequency of node j to the local clock frequency of node i. Thus, from Equations (4) and (7) we have:

$$\mathcal{C}_j(t_1) = \hat{\alpha}_{ij}\tau_i(t_1) + \hat{\beta}_{ij}$$

Subtracting $\tau_i(t_1)$ from each side of this equation leads to:

$$\mathcal{C}_j(t_1) - \tau_i(t_1) = \hat{\alpha}_{ij}\tau_i(t_1) - \tau_i(t_1) + \hat{\beta}_{ij}$$

As a result:

$$\mathit{offset}_{ij}^{t_1} = (\hat{\alpha}_{ij} - 1)\tau_i(t_1) + \hat{\beta}_{ij} \tag{17}$$

Similarly, at time t_2 we have:

$$\mathit{offset}_{ij}^{t_2} = (\hat{\alpha}_{ij} - 1)\tau_i(t_2) + \hat{\beta}_{ij} \tag{18}$$

and subtracting (17) from (18) gives:

$$\mathit{offset}_{ij}^{t_2} - \mathit{offset}_{ij}^{t_1} = (\hat{\alpha}_{ij} - 1)(\tau_i(t_2) - \tau_i(t_1)) \tag{19}$$

Consequently,

$$\alpha_{ij} = \frac{\mathit{offset}_{ij}^{t_2} - \mathit{offset}_{ij}^{t_1}}{\tau_i(t_2) - \tau_i(t_1)}$$

□

Appendix B Proof of Theorem 1 (see page 9)

Proof According to Lemma 1 and Equation (11), we have:

$$\begin{aligned}
\alpha_i(k+1) &= \sum_{j\in\mathcal{N}_i} w_{ij}(k) \left(\frac{\mathit{offset}_{ij}^{t_2} - \mathit{offset}_{ij}^{t_1}}{\tau_i(t_2) - \tau_i(t_1)} \right)(k) \\
&= \sum_{j\in\mathcal{N}_i} w_{ij}(k) \left(\frac{(\mathcal{C}_j(t_2) - \tau_i(t_2)) - (\mathcal{C}_j(t_1) - \tau_i(t_1))}{\tau_i(t_2) - \tau_i(t_1)} \right)(k) \\
&= \sum_{j\in\mathcal{N}_i} w_{ij}(k) \left(\frac{\mathcal{C}_j(t_2) - \mathcal{C}_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} - 1 \right)(k)
\end{aligned}$$

By adding 1 to each side of the equation, we obtain:

$$\alpha_i(k+1) + 1 = 1 + \sum_{j\in\mathcal{N}_i} w_{ij}(k) \left(\frac{\mathcal{C}_j(t_2) - \mathcal{C}_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} - 1 \right)(k)$$

As $\hat{\alpha}_i = \alpha_i + 1$, the above equation can be written as:

$$\hat{\alpha}_i(k+1) = 1 + \sum_{j \in \mathcal{N}_i} w_{ij}(k) \left(\frac{\mathcal{C}_j(t_2) - \mathcal{C}_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} - 1 \right)(k)$$

From Equation (12), we can easily prove that $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$, implying:

$$\begin{aligned} \hat{\alpha}_i(k+1) &= \sum_{j \in \mathcal{N}_i} w_{ij}(k) + \sum_{j \in \mathcal{N}_i} w_{ij}(k) \left(\frac{\mathcal{C}_j(t_2) - \mathcal{C}_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} - 1 \right)(k) \\ &= \sum_{j \in \mathcal{N}_i} w_{ij}(k) \left(\frac{\mathcal{C}_j(t_2) - \mathcal{C}_j(t_1)}{\tau_i(t_2) - \tau_i(t_1)} \right)(k) \end{aligned}$$

According to (3) and (8), we can substitute the values of both $\mathcal{C}_j(t)$ and $\tau_i(t)$ into the above equation to obtain:

$$\begin{aligned} \hat{\alpha}_i(k+1) &= \sum_{j \in \mathcal{N}_i} w_{ij}(k) \left(\frac{(\hat{\alpha}_j a_j t_2 + \hat{\alpha}_j b_j + \hat{\beta}_j) - (\hat{\alpha}_j a_j t_1 + \hat{\alpha}_j b_j + \hat{\beta}_j)}{(a_i t_2 + b_i) - (a_i t_1 + b_i)} \right)(k) \\ &= \sum_{j \in \mathcal{N}_i} w_{ij}(k) \frac{\hat{\alpha}_j a_j}{a_i}(k) \end{aligned}$$

If we multiply each side of the equation by a_i we obtain:

$$\hat{\alpha}_i a_i(k+1) = \sum_{j \in \mathcal{N}_i} w_{ij}(k) \hat{\alpha}_j a_j(k)$$

If we denote $\hat{\alpha}_i a_i$ by x_i, we can rewrite the above equation as follows:

$$x_i(k+1) = \sum_{j \in \mathcal{N}_i} w_{ij}(k) x_j(k)$$

which can be rewritten more compactly as:

$$\mathbf{x}(k+1) = \mathcal{W}(k)\mathbf{x}(k)$$

where $\mathcal{W}$ denotes the $n \times n$ matrix whose elements are the weights w_{ij} defined by Equation (12), and where $\mathbf{x}(k) = [x_1(k), ..., x_n(k)]^T$ defines a column vector whose elements represent the clock rates of all compensated clocks of the network at time slot k. According to Equation (12), the elements of $\mathcal{W}$ are nonnegative. In addition, $\mathcal{W}$ is row-stochastic since it is nonnegative and the sum of the elements of each row is equals 1, i.e., $\mathcal{W}\mathbf{1} = \mathbf{1}$ where $\mathbf{1} = (1, 1, 1, ..., 1)^T$. Also, the state of each node $i \in \mathcal{V}$ will be influenced directly or indirectly by every node $j \in \mathcal{V}$ through a sequence of communications as $\mathcal{G}$ is connected according to Assumption 1. Following [28, 29], all this provides a sufficient condition to guarantee that all x_i will converge to a steady-state value $\mathbf{c}$:

$$\lim_{k \to \infty} \mathbf{x}(k) = \mathbf{c1}$$

Consequently, as $x_i = \hat{\alpha}_i a_i$,

$$\lim_{k \to \infty} \hat{\alpha}_i(k) a_i = \alpha_v \mathbf{1}$$

□

Appendix C Proof of Lemma 2 (see page 10)

Proof According to Equation (19), we can write:

$$\begin{aligned} \mathit{offset}_{ij}^{t} - \mathit{offset}_{ij}^{old} &= (\hat{\alpha}_{ij} - 1)(\tau_i(t) - \tau_i(t_{old})) \\ \mathit{offset}_{ij}^{t} &= \mathit{offset}_{ij}^{old} + (\hat{\alpha}_{ij} - 1)(\tau_i(t) - \tau_i(t_{old})) \\ \mathcal{C}_j(t) - \tau_i(t) &= \mathit{offset}_{ij}^{old} + (\hat{\alpha}_{ij} - 1)(\tau_i(t) - \tau_i(t_{old})) \end{aligned}$$

Consequently,

$$\mathcal{C}_j(t) = \hat{\alpha}_{ij}\tau_i(t) + \mathit{offset}_{ij}^{old} - (\hat{\alpha}_{ij} - 1)\tau_i(t_{old}) \tag{20}$$

□

Appendix D Proof of Theorem 2 (see page 10)

Proof The Equation (14) can be written as:

$$\mathcal{C}_i(t) = \hat{\alpha}_i\overline{a}_i t + \hat{\alpha}_i\overline{b}_i + \mathit{offset}_i^{old} - (\hat{\alpha}_i - 1)\overline{\tau}_i(t_{old}) \tag{21}$$

where $\overline{a}_i t = \sum_{j\in\mathcal{N}_i} w_{ij}a_i t$ and $\overline{b}_i = \sum_{j\in\mathcal{N}_i} w_{ij}b_i$ as stated by Equation (16). According to Equations (8) and (21), we have:

$$\hat{\alpha}_i(k+1)b_i + \hat{\beta}_i(k+1) = \hat{\alpha}_i(k)\overline{b}_i + \mathit{offset}_i^{old}(k) - (\hat{\alpha}_i - 1)\overline{\tau}_i(t_{old})(k)$$

We denote $\hat{\alpha}_i(k+1)b_i + \hat{\beta}_i(k+1)$ by $x_i(k+1)$ to obtain:

$$x_i(k+1) = \hat{\alpha}_i(k)\overline{b}_i + \mathit{offset}_i^{old}(k) - (\hat{\alpha}_i - 1)\overline{\tau}_i(t_{old})(k)$$

Using Equations (15) and (16), we can substitute the values of both offset_i^{old} and $\overline{\tau}_i(t_{old})$ into the above equation to obtain:

$$\begin{aligned} x_i(k+1) &= \hat{\alpha}_i(k)\overline{b}_i + \sum_{j\in\mathcal{N}_i} w_{ij}\,\mathit{offset}_{ij}(k) - (\hat{\alpha}_i - 1)\sum_{j\in\mathcal{N}_i} w_{ij}\tau_i(t)(k) \\ &= \hat{\alpha}_i(k)\overline{b}_i + \sum_{j\in\mathcal{N}_i} w_{ij}(\mathcal{C}_j(t) - \tau_i(t) - (\hat{\alpha}_i - 1)\tau_i(t))(k) \end{aligned}$$

According to Equation (7), we can substitute the value of $\mathcal{C}_j(t)$ into the above equation to obtain:

$$\begin{aligned}
x_i(k+1) &= \hat{\alpha}_i(k)\bar{b}_i + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j\tau_j(t) + \hat{\beta}_j - \hat{\alpha}_i\tau_i(t))(k) \\
&= \hat{\alpha}_i(k)\bar{b}_i + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j a_j t + \hat{\alpha}_j b_j + \hat{\beta}_j - \hat{\alpha}_i a_i t - \hat{\alpha}_i b_i)(k) \\
&= \hat{\alpha}_i(k)\bar{b}_i - \hat{\alpha}_i(k)\sum_{j\in\mathcal{N}_i} w_{ij}b_i + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j a_j t - \hat{\alpha}_i a_i t)(k) + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j b_j + \hat{\beta}_j)(k) \\
&= \hat{\alpha}_i(k)\bar{b}_i - \hat{\alpha}_i(k)\bar{b}_i + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j a_j t - \hat{\alpha}_i a_i t)(k) + \sum_{j\in\mathcal{N}_i} w_{ij}x_j(k) \\
&= \sum_{j\in\mathcal{N}_i} w_{ij}x_j(k) + \sum_{j\in\mathcal{N}_i} w_{ij}(\hat{\alpha}_j a_j t - \hat{\alpha}_i a_i t)(k)
\end{aligned}$$

which can be rewritten more compactly as:

$$\mathbf{x}(k+1) = \mathcal{W}(k)\mathbf{x}(k) + \boldsymbol{\phi}(k)$$

According to Theorem 1, $\lim_{k\to\infty} a_i\hat{\alpha}_i(k) = \alpha_v, \forall i \in \mathcal{V}$, and thus $\lim_{k\to\infty} \boldsymbol{\phi}(k) = 0$. Likewise, $\mathcal{W}$ is row-stochastic since it is nonnegative and the sum of the entries of each row equals 1. Also, according to Assumption 1, each node $i \in \mathcal{V}$ can communicate its values to every other node $j \in \mathcal{V}$. As a result [28, 29], x_i will converge to a steady-state value $\mathbf{c}$:

$$\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{c1}$$

Consequently, as $x_i = \hat{\alpha}_i b_i + \hat{\beta}_i$, we obtain:

$$\lim_{k\to\infty} \hat{\alpha}_i(k)b_i + \hat{\beta}_i(k) = \beta_v\mathbf{1}$$

□

Acknowledgements This paper has been written while the first author was visiting the Wireless Sensor group at Lab-STICC in Brest. Financial support through the PNE program established by the Government of Algeria is gratefully acknowledged.

References

1. D. Mills. Internet time synchronization: the network time protocol. In *IEEE Transactions on Communications*, 1991.
2. Jeremy Elson and Kay Römer. Wireless sensor networks: A new regime for time synchronization. *ACM SIGCOMM Computer Communication Review*, 33(1):149–154, 2003.
3. Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 2002.

4. Saurabh Ganeriwal, Ram Kumar, and Mani B Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, 2003.
5. Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *ACM Second Intl Conference on Embedded Networked Sensor Systems (SenSys 04)*, pages 39–49, 2004.
6. Michael Kevin Maggs, Steven G O'Keefe, and David Victor Thiel. Consensus clock synchronization for wireless sensor networks. *Sensors Journal, IEEE*, 12(6):2269–2277, 2012.
7. Philipp Sommer and Roger Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 37–48. IEEE Computer Society, 2009.
8. Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. Pulsesync: an efficient and scalable clock synchronization protocol. *Networking, IEEE/ACM Transactions on*, 2015.
9. Luca Schenato and Federico Fiorentin. Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 47(9):1878–1886, 2011.
10. Jianping He, Peng Cheng, Ling Shi, Jiming Chen, and Youxian Sun. Time synchronization in wsns: A maximum-value-based consensus approach. *Automatic Control, IEEE Transactions on*, 59(3):660–675, 2014.
11. Branislav Kusy, Prabal Dutta, Philip Levis, Miklos Maroti, Akos Ledeczi, and David Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. *International Journal of Ad Hoc and Ubiquitous Computing*, 1(4):239–251, 2006.
12. Philipp Sommer and Roger Wattenhofer. Symmetric clock synchronization in sensor networks. In *Proceedings of the workshop on Real-world wireless sensor networks*, pages 11–15. ACM, 2008.
13. Yin Chen, Roberto Tron, Andreas Terzis, and Rene Vidal. Accelerated corrective consensus: Converge to the exact average at a faster rate. In *American Control Conference (ACC), 2011*, pages 3417–3422. IEEE, 2011.
14. Boris N Oreshkin, Tuncer C Aysal, and Mark J Coates. Distributed average consensus with increased convergence rate. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 2285–2288. IEEE, 2008.
15. C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 225–238, 2009.
16. Jianping He, Peng Cheng, Ling Shi, Jiming Chen, and Youxian Sun. Time synchronization in wsns: a maximum-value-based consensus approach. *IEEE Transactions on Automatic Control*, 59(3):660–675, 2014.
17. Jie Wu, Liyi Zhang, Yu Bai, and Yunshan Sun. Cluster-based consensus time synchronization for wireless sensor networks. *Sensors Journal, IEEE*, 15(3):1404–1413, 2015.

18. Wendi B Heinzelman, Anantha P Chandrakasan, and Hari Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, 2002.
19. Zhe Yang, Liang He, Lin Cai, and Jianping Pan. Temperature-assisted clock synchronization and self-calibration for sensor networks. *IEEE Transactions on Wireless Communications*, 13(6):3419–3429, 2014.
20. Miao Xu and Wenyuan Xu. Taco: Temperature-aware compensation for time synchronization in wireless sensor networks. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 122–130. IEEE, 2013.
21. Chafika Benzaïd, Miloud Bagaa, and Mohamed Younis. Efficient clock synchronization for clustered wireless sensor networks. *Ad Hoc Networks*, 2016.
22. Djamel Djenouri, Nassima Merabtine, Fatma Zohra Mekahlia, and Messaoud Doudou. Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks. *Ad Hoc Networks*, 11(8):2329–2344, 2013.
23. Bin Luo, Lei Cheng, and Yik-Chung Wu. Fully distributed clock synchronization in wireless sensor networks under exponential delays. *Signal Processing*, 125:261–273, 2016.
24. Pierre-Alexandre Bliman and Giancarlo Ferrari-Trecate. Average consensus problems in networks of agents with delayed communications. *Automatica*, 44(8):1985–1995, 2008.
25. Ruggero Carli. Topics on the average consensus problems. 2008.
26. Konstantin Avrachenkov, Mahmoud El Chamie, and Giovanni Neglia. A local average consensus algorithm for wireless sensor networks. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–6. IEEE, 2011.
27. F Knorn, R Stanojevic, M Corless, and R Shorten. A framework for decentralised feedback connectivity control with application to sensor networks. *International Journal of Control*, 82(11):2095–2114, 2009.
28. Ming Cao, A Stephen Morse, and Brian DO Anderson. Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008.
29. Alex Olshevsky and John N Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48 (1):33–55, 2009.
30. F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network, IEEE*, 18(4):45–50, July 2004.
31. Jennifer Lundelius and Nancy Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2–3):190 – 204, 1984.
32. S. Graham and P.R. Kumar. Time in general-purpose control systems: the control time protocol and an experimental evaluation. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 4, pages 4004–4009 Vol.4, 2004.

33. Athanassios Boulis. Castalia: revealing pitfalls in designing distributed algorithms in wsn. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 407–408. ACM, 2007.
34. Dimosthenis Pediaditakis, Yuri Tselishchev, and Athanassios Boulis. Performance and scalability evaluation of the castalia wireless sensor network simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, page 53. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.
35. Fengyuan Ren, Chuang Lin, and Feng Liu. Self-correcting time synchronization using reference broadcast in wireless sensor network. *IEEE Wireless Communications*, 15(4), 2008.
36. SS Hussain and P Sprent. Non-parametric regression. *Journal of the Royal Statistical Society. Series A (General)*, pages 182–191, 1983.