A Flexible Linear Control Algorithm for Wireless Sensor Network Synchronization

Daniele Fontanelli, David Macii and Dario Petri
Department of Information Engineering and Computer Science
University of Trento
Via Sommarive 14, 38100, Trento, Italy
Email: {fontanelli,macii,petri}@disi.unitn.it

***Abstract*—Time synchronization of wireless sensor network (WSN) nodes is essential to schedule orderly communication and monitoring tasks. In this paper, a synchronization algorithm based on linear control theory is used to compensate at run–time both time offsets and clock frequency skews. The proposed approach is innovative for two reasons. First of all, the controller correcting the local timers is able to work also in nonideal conditions (e.g., including random communication latencies and numerical quantization noise). Secondly, the computational burden of the algorithm and the amount of network traffic to transfer timing data are compatible with the requirements of typical low–cost WSN nodes. In fact, the time intervals between subsequent synchronizations tend to grows while the uncertainty decreases. As a consequence, each node is increasingly free to run other tasks. The correct operation of the proposed algorithm has been proved in a simulation framework modeling with great accuracy the behavior of a cluster of nodes.**

I. Introduction

The recent deployment of wireless sensor networks (WSNs) in a variety of applications, ranging from environmental monitoring services to factory automation, has promoted the development of different synchronization protocols aimed at coordinating the tasks running on different nodes of the network. While addressing this problem is certainly essential when strict real–time requirements are necessary, synchronization is beneficial also to reduce the energy consumption whenever the network nodes have to be active just for a limited fraction of time. In fact, by synchronizing the wake–up and sleep times of the radio module, unnecessary data transfers could be avoided. The problem of time synchronization has been deeply investigated in last years, but it is still of paramount interest for the scientific communities dealing with networking and measurement science (e.g., to estimate the distance between different devices [1]). For instance, a new version of the well–known Precision Time Protocol (PTP) for distributed measurement and control systems has been recently released [2]. Although this protocol is general enough to be adapted to different network types, it is not very suitable for WSNs, due to its considerable communication overhead and to the major differences between the characteristics of PTP and those of the lower level communication protocols commonly used in WSNs (especially the standard IEEE 802.15.4 [3]). This is the reason why several WSN–specific synchronization protocols have been proposed in last years. Such protocols are usually able to achieve just moderate accuracy, but with low computational burden and little network traffic [4]. In fact, it is often preferable to decrease the synchronization rate, while keeping node timers within given tolerance boundaries that in turn depend on the requirements of the target application [5], [6]. In general, a node elected (or dynamically re–elected) as the time reference of the network transfers its local time to the other devices (possibly through multiple hops), while compensating the clock offsets between parent and child nodes [7], [8].

In this paper, we propose a different approach for synchronization. Instead of using a single time reference, each node computes and updates individually its own local time on the basis of the temporal information collected from all nearby nodes. The technique for exchanging local time data is similar to the mechanism underlying the Reference Broadcast Synchronization (RBS) protocol proposed by Elson in [9]. On the other hand, the clock correction algorithm relies on an enhanced version of the proportional and integral (PI) consensus controller originally described in [10] and further analyzed by Carli et al. in [11]. This algorithm has been suitably changed in order to address serious practical problems such as random communication latencies and timing quantization issues. Moreover, the algorithm has been designed to reduce progressively the number of synchronization events while accuracy improves.

In the following, at first in Section II the problem of clock synchronization is properly formulated. Then, in Section III the principle of operation of the controller correcting the local clocks is described and justified theoretically. In Section IV the basic steps of the synchronization algorithm are described. Finally, in Section V some simulation results are reported.

II. Problem Formulation

Let us consider a generic WSN consisting of n identical nodes, each one with a unique identifier in the set $ID = \{1, \ldots, n\}$. As known, the relative time of each node can be measured by means of a local timer clocked by a local (preferably crystal) oscillator of nominal frequency f_0. Formally, the WSN timers can be regarded as a set of discrete–time linear integrators, i.e.,

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{d}(t) + \underline{\mathbf{q}}(t), \qquad (1)$$

where

- $t \in \mathbb{N}_0$ represents the number of clock ticks on an ideal (i.e., perfect) timescale;
- $\mathbf{x} \in \mathbb{R}^n$ is the column vector containing the time values of all WSN nodes and formally correspond to the state of the system;
- $\underline{\mathbf{q}} \in \mathbb{R}^n$ is the random vector (whose elements are uniformly distributed in the interval $[0 \frac{1}{f_0}]$) modeling the quantization noise due to the finite resolution of the digital timer[1];
- $\mathbf{d} \in \mathbb{R}^n$ is the vector containing the actual time increments of each clock during the $t-$th tick of the timer.

In particular, $\mathbf{d}$ is given by:

$$\mathbf{d}(t) = \frac{1}{f_0}\mathbf{1} + \boldsymbol{\Delta}(t) + \underline{\boldsymbol{\nu}}(t) \tag{2}$$

where $\mathbf{1}$ is the unit vector (i.e., with all entries equal to 1), $\boldsymbol{\Delta} \in \mathbb{R}^n$ is the vector of the systematic offsets between the ideal and the real clock periods of the various nodes and $\underline{\boldsymbol{\nu}} \in \mathbb{R}^n$ is the random vector modeling the jitter resulting from the superimposition of different types of power–law noise [12]. Although the systematic contributions may change in time due to aging and variable operating conditions, in the following the elements of $\boldsymbol{\Delta}$ will be considered to be approximately constant because their dynamics is negligible in time intervals in the order of several minutes. Of course, if the initial values of the local timers are the same [i.e. $x_i(0) = x_j(0)$] and if $d_i(0) = 1/f_0$, for all $i, j = 1, \ldots, n$, then the clocks are synchronized from the very beginning and they will remain synchronized. However, in a realistic scenario initial clock offsets and clock frequency skews differ from node to node, thus leading the corresponding time values to drift away from one another.

In this context, the term *time synchronization* should be intended as the compensation of the inter–node time differences, regardless of the Coordinated Universal Time (UTC). Hence, the nodes can be considered as *synchronized*, if there exists a finite time $\bar{t}$ and two values $b_1, b_2 \in \mathbb{R}$ such that $\|x_i(t) - (b_1 t + b_2)\| \leq \varepsilon$ for $t \geq \bar{t}$ and for $\forall i = 1, \ldots, n$. Notice that ε can not be smaller than timer resolution, i.e. $\epsilon \geq 1/f_0$.

In principle, we can achieve time synchronization by suitably controlling the timers of the various nodes. Accordingly, the discrete–time linear system (1) can be modified as follows:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{d}(t) + \underline{\mathbf{q}}(t) + \mathbf{u}(t), \tag{3}$$

with $\mathbf{u} \in \mathbb{R}^n$ being the vector of the control inputs to each timer. In [11] it is proved that the synchronization problem can be solved by using a *Proportional Integral* (PI) controller on each node. In matrix notation, a general expression for this controller is:

$$\mathbf{y}(t+1) = \mathbf{y}(t) - \alpha \mathbf{K} \underline{\hat{\mathbf{x}}}(t) \tag{4}$$

$$\mathbf{u}(t) = \mathbf{y}(t) - \mathbf{K} \underline{\hat{\mathbf{x}}}(t) \tag{5}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the state vector of the controller, the feedback matrix $\mathbf{K}$ and the coefficient α result from the consensus–related theory (see [10]) and $\underline{\hat{\mathbf{x}}} = \mathbf{x} - \underline{\boldsymbol{l}}$ is the vector of the time values measured by the various nodes. Notice that in general the elements of $\underline{\hat{\mathbf{x}}}$ are different from those of $\mathbf{x}$ because both collecting the time values from all nodes and computing the next controller output requires some additional time. Such latencies are represented by the random vector $\underline{\boldsymbol{l}} \in \mathbb{R}^n$ and are quite significant because they can be in the order of several tens of ms (i.e., $\underline{\boldsymbol{l}} \geq 0$). Also, the communication latencies may increase considerably as a function of the number of nodes for two reasons. Firstly, the number of messages for transferring the time values of each node to all the other devices grows linearly with n. Secondly, both the probability of sensing the channel busy and the probability of having packet collisions also increase, thus causing multiple retransmission attempts, with major increments in terms of delay [13]. The main consequence of this observation is that the input values to the controller can be updated at a rate which is much smaller than the frequency of the oscillator clocking the timer. Nonetheless, this problem can be addressed by properly designing the controller as it will be explained in the next section.

III. Controller Design

The PI controller design relies on the assumption that the controller state as well as the corresponding output vector can be updated only when each node receives successfully the time values measured by all the other nodes. Such a hypothesis is particularly important because it implies that the controller switches between two different configurations.

The first one is represented by equations (4) and (5) and occurs as soon as a new complete set of local time values for all nodes is available. In this case, plugging the equation (5) into clock dynamic equation (3), we have

$$\begin{aligned} \begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{y}(t+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_n - \mathbf{K} & \mathbf{I}_n \\ -\alpha\mathbf{K} & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}(t) \end{bmatrix} + \boldsymbol{\xi}^1(t) \\ &= \mathbf{A}^1_{cl} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}(t) \end{bmatrix} + \boldsymbol{\xi}^1(t)\,, \end{aligned} \tag{6}$$

where $\mathbf{I}_n$ is a $n \times n$ identity matrix and

$$\boldsymbol{\xi}^1(t) = \begin{bmatrix} \mathbf{d}(t) + \underline{\mathbf{q}}(t) + \mathbf{K}\underline{\boldsymbol{l}} \\ \alpha\mathbf{K}\underline{\boldsymbol{l}} \end{bmatrix},$$

is the vector containing all the described nuisances.

The second configuration instead corresponds to the case in which a complete set of new local time values $\underline{\hat{\mathbf{x}}}$ is not available at time t. This may happen because the data collection process between two subsequent synchronizations is still ongoing or because some data transfer fails. In such cases, the safest policy is to hold constant the output $\mathbf{u}$ of the controller. This, in practice, is equivalent to set $\underline{\hat{\mathbf{x}}}$ in (4) and (5) equal to zero. Hence, the corresponding closed–loop dynamics is

$$\begin{aligned} \begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{y}(t+1) \end{bmatrix} &= \begin{bmatrix} \mathbf{I}_n & \mathbf{I}_n \\ 0 & \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}(t) \end{bmatrix} + \boldsymbol{\xi}^0(t) \\ &= \mathbf{A}^0_{cl} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{y}(t) \end{bmatrix} + \boldsymbol{\xi}^0(t)\,, \end{aligned} \tag{7}$$

[1]In this paper random variables are denoted by underlined symbols.

where
$$\boldsymbol{\xi}^0(t) = \begin{bmatrix} \mathbf{d}(t) + \underline{\mathbf{q}}(t) \\ 0 \end{bmatrix}.$$

The system switches between the two different dynamics depend on the time interval between two subsequent synchronizations. The lower bound $\underline{\gamma}$ of this interval is equal to the (random) time spent to run each iteration of the synchronization procedure. On the contrary, the time interval $\gamma_k \geq \underline{\gamma}$ after the $k-$th synchronizations can be made, surprisingly, arbitrarily large.

In more strict theoretic terms, the overall closed loop is governed by a *switching signal* $\sigma(t, \gamma_k)$
$$\sigma(t, \gamma_k) = \begin{cases} 1, & \text{if local time values are available} \\ 0, & \text{otherwise} \end{cases}$$
Therefore, if we define $\mathbf{A}_{cl}^{\sigma(t,\gamma_k)}$ as the closed–loop matrix, the dynamics of the system is given by
$$\overbrace{\mathbf{A}_{cl}^0 \cdots \mathbf{A}_{cl}^0 \mathbf{A}_{cl}^1}^{\gamma_k} = \mathbf{A}_{cl}^{0\,\gamma_k - 1} \mathbf{A}_{cl}^1 =$$
$$\mathbf{A}_{cl_{\gamma_k}} = \begin{bmatrix} \mathbf{I}_n - [1 + \alpha(\gamma_k - 1)]\mathbf{K} & \gamma_k \mathbf{I}_n \\ -\alpha \mathbf{K} & \mathbf{I}_n \end{bmatrix}. \tag{8}$$

It is worthwhile to note that, with appropriate choices of the parameter α and the feedback matrix $\mathbf{K}$, the system can be made stable and
$$x_i(t) \rightarrow \frac{1}{n} \sum_{j=1}^{n} [d_j t + x_j(0)], \ \forall i = 1, \ldots, n, \ \ t \rightarrow +\infty. \tag{9}$$
In other words, every timer converges to the mean of the time values measured by all network nodes. Indeed, this is the basic idea behind the consensus problems in the control literature as mentioned in [10], [11]. In such a way, the clock synchronization solution is turned into an asymptotic stability problem, i.e., $\mathbf{x}(t) - (b_1 t + b_2)\mathbf{1} \rightarrow 0$, for $t \rightarrow +\infty$. Notice that the vector $\boldsymbol{\xi}^1(t)$ and $\boldsymbol{\xi}^0(t)$ represent unknown, but bounded, quantities (i.e., additional inputs to the linear closed–loop system). Hence, the asymptotic stability of the overall system is only given by the eigenvalues of the dynamic matrix $\mathbf{A}_{cl_{\gamma_k}}$. In practice, the effect of $\boldsymbol{\xi}^1(t)$ and $\boldsymbol{\xi}^0(t)$ is to decrease the steady state accuracy of the synchronization algorithm.

The asymptotic stability of the system governed by the dynamic matrix (8) for a fixed γ_k can be obtained by choosing the eigenvalues λ_i of $\mathbf{K}$, with $i = 1, \ldots, n$, in the set $(0, 4/[2 + \alpha(\gamma_k - 2)])$, where $\alpha \in (0, 1)$. Furthermore, the matrix $\mathbf{K}$ must be symmetric and such that $\mathbf{K1} = 0$ for the reasons stated in [10]. The choice of the matrix $\mathbf{K}$ is related to the *Laplacian* of the *visibility matrix*, also known as *adjacency matrix*. This matrix, in the following referred to as $\mathbf{V}$, is related to the undirected graph describing the visibility between the WSN nodes. In particular, the entry (i, j) of $\mathbf{V}$ is set equal to 1 if node i is able "to see" node j and vice versa. The Laplacian of $\mathbf{V}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{V}$, where $\mathbf{D}$ is the so–called *degree matrix*, namely the diagonal matrix containing the number of connections of each node. It can be proved that the $\mathbf{L}$ is a symmetric matrix with eigenvalues lying in the interval $[0, 2]$. Moreover, $\mathbf{L1} = 0$ [10]. Notice that $\mathbf{L}$ can be easily computed if the visibility matrix is known. The feedback matrix $\mathbf{K}$ is then obtained by suitably scaling the Laplacian matrix. In the case of complete visibility, i.e., when all the WSN nodes of a cluster are able to communicate within a single–hop link, the feedback matrix $\mathbf{K}$ and the parameter α can be computed by each node on the basis of the planned synchronization period γ_k. In particular, by choosing
$$\alpha = \frac{1}{\gamma_k + 1} \tag{10}$$
and
$$\mathbf{K} = \frac{\gamma_k + 1}{n\gamma_k}(\mathbf{D} - \mathbf{V}), \tag{11}$$
two of the eigenvalues of the closed–loop matrix in (8) are equal to 1 and all the others are 0, which guarantees the fastest convergence. Indeed, for a stable discrete system, the rate of convergence is given by the largest eigenvalue that differs from 1. The smaller the norm of the eigenvalue, the faster the convergence rate is.

As stated above, due to the intrinsic latencies associated to data collection and processing times, the closed–loop system switches between two different controllers, described by the single closed–loop matrix in (8). If the time interval γ_k between the $k-$th and the $(k{+}1)-$th synchronization changes, then also (8) must change according to (10) and (11). This leads to a set of switching matrices, namely one for each γ_k. Even if all closed–loop systems are asymptotically stable for a fixed γ_k, the stability of the overall switching system for different values of γ_k is not guaranteed. In fact, switching between asymptotically stable systems may result in an overall unstable behavior [14]. In order to prevent possible instabilities, the system should not switch during the transient between two different dynamics [15]. In particular, the system should wait a proper *dwell time* between two successive switches. In practice, if we define $\sigma_x^2(t)$ as the mean square error (MSE) of the time differences for the first time t in which the period γ_k is selected, then the system is asymptotically stable (i.e., $\sigma_x^2(t)$ tends to zero). However, if $\gamma_{k+1} \neq \gamma_k$ at a certain time $\bar{t} > t$ and $\sigma_x^2(t) \leq \sigma_x^2(\bar{t})$, then the next synchronization period has to be kept constant, namely still equal to γ_k. Conversely, if $\sigma_x^2(t) > \sigma_x^2(\bar{t})$, the next synchronization period can be safely set to γ_{k+1}. This way, the asymptotic stability of the switching system is assured and the synchronization uncertainty still globally decreases, although some local divergence during the dwell time may exist.

IV. Synchronization Procedure

The synchronization procedure consists of four iterative steps or phases, which are described in the following. The first iteration of the procedure is slightly different from the others, because no timer correction are performed. Basically, the first iteration is used to estimate the duration of the procedure as well as the MSE ε_1 of the initial synchronization errors. Accordingly, γ_1 is set slightly larger than the procedure duration. In fact, γ_1 must be as small as possible to speed up the synchronization process.

A. Beacon Packet Broadcasting

When a generic synchronization interval γ_k (for any $k \geq 1$) expires, the node with identifier equals to:

$$m = \begin{cases} \text{mod}(k,n) & \text{mod}(k,n) \neq 0 \\ n & \text{mod}(k,n) = 0 \end{cases} \tag{12}$$

(in the following referred to as *synchronization master* or SM) broadcasts a time–stamped *beacon packet* containing just its unique identifier, the current synchronization interval γ_k and the value of ε_k. In general, all time–stamping operations should be performed at the Media Access Control (MAC) layer (i.e., just before being transmitted), in order to minimize the latencies due to packet formatting, data transfer and channel access [16]. In the following, we will refer to t_{s_m} as the sending time–stamp of the synchronization beacon. Each node $i \neq m$ receiving the synchronization beacon time–stamps the incoming packet using its local clock at time $t^i_{r_m}$ and it sets to 1 the element (m, i) of the visibility matrix $\mathbf{V}$. Of course, t_{s_m} differs from $t^i_{r_m}$ because of the communication latency. On the other hand, the differences between the values of $t^i_{r_m}$ are mainly caused by unequal initial conditions of the timers and/or by different clock drifts. In fact, if the WSN nodes are located quite close to one another (i.e., within no more than some tens of meters), the send and communication delays associated with beacon broadcasting are approximately the same for all nodes, while the time to run the interrupt service routines reading the timers in receiving devices is much smaller than the average communication latency.

B. Acknowledgment Packet Broadcasting

In the second phase of the procedure, each node with $i \neq m$ broadcasts the value of its own beacon time–stamp within an *acknowledgment packet*, which in turn is also time–stamped at time t_{s_i}. Acknowledgment packets are transmitted strictly in sequential order. This means that the $i-$th node broadcasts its own packet only after receiving the acknowledgement packet sent by the $(i-1)-$th node or after a preset maximum time–out interval expires. Any node with $j \neq i$ that receives the acknowledgment packet from the $i-$th device also time–stamps the incoming packet at time $t^j_{r_i}$ and sets to 1 the entry (i, j) of the visibility matrix.

C. Controller Update and Timer Correction

Once the acknowledgment broadcasting phase is completed, every node $i = 1, \ldots, n$ contains just the $i-$th row $\mathbf{V}_i$ of the visibility matrix. Accordingly, node i computes its degree of connectivity by summing up the elements of $\mathbf{V}_i$ and it also builds the $i-$th row $\mathbf{D}_i$ of the degree matrix. Afterwards, all WSN nodes run the following sequence of operations, i.e.:

- At first, they compute the rows of the feedback matrix related to the synchronization interval γ_k as described in Section III, i.e.

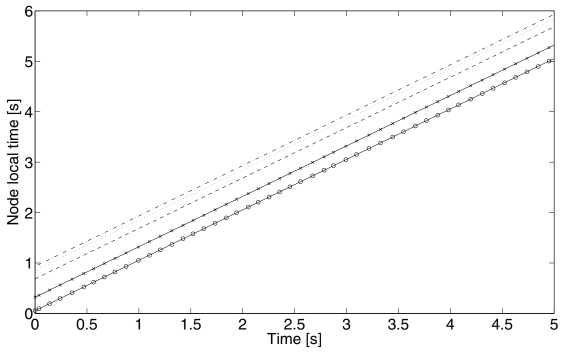
$$\mathbf{K}^k_i = \frac{\gamma_k + 1}{n\gamma_k}(\mathbf{D}_i - \mathbf{V}_i) \quad i = 1, \ldots, n \tag{13}$$

- Then, each device $i = 1, \ldots, n$ estimates the mean communication latency as the average of all the differences between the reception and sending time–stamps collected by the node considered. In particular, the average communication latency is given by:

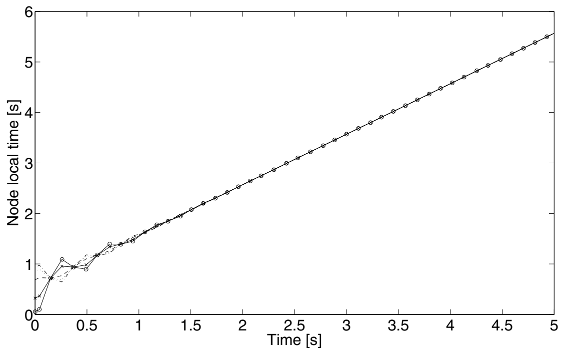
$$\bar{\delta}_i = \begin{cases} \frac{1}{n-1}\left[\sum_{\substack{j=1 \\ j\neq m}}^{n} \frac{(t^j_{r_m} - t_{s_m}) + (t^m_{r_j} - t_{s_j})}{2}\right] & i = m \\ \frac{1}{n-1}\left[\frac{(t^i_{r_m} - t_{s_m}) + (t^m_{r_i} - t_{s_i})}{2} + \sum_{\substack{j=1 \\ j\neq i,m}}^{n} \frac{(t^i_{r_j} - t_{s_j}) + (t^j_{r_i} - t_{s_i})}{2}\right] & i \neq m \end{cases} \tag{14}$$

- The values of (14) are used to compensate the differences between the reception time–stamps of the beacon packet and the sending time–stamp applied by the SM. Accordingly, node i computes the column vector $\underline{\hat{\mathbf{x}}}_i = [t^1_{r_m}, \ldots, t_{s_m} + \bar{\delta}_i \ldots, t^n_{r_m}]^T$.
- Finally, the state value and the output of the controller of each node result from:

$$y_i(t+1) = y_i(t) - \alpha \mathbf{K}^k_i \cdot \underline{\hat{\mathbf{x}}}_i(t) \tag{15}$$
$$u_i(t) = y_i(t) - \mathbf{K}^k_i \cdot \underline{\hat{\mathbf{x}}}_i(t) \tag{16}$$



(a)



(b)

Fig. 1. Time values measured by 5 WSN nodes before (a) and after (b) running the synchronization procedure. Each line style corresponds to a different node. In both cases, the initial time offsets are the same and are randomly chosen in the range $[0, 1]$ s. In (b) all timers converge to the mean of the time values measured by all nodes.

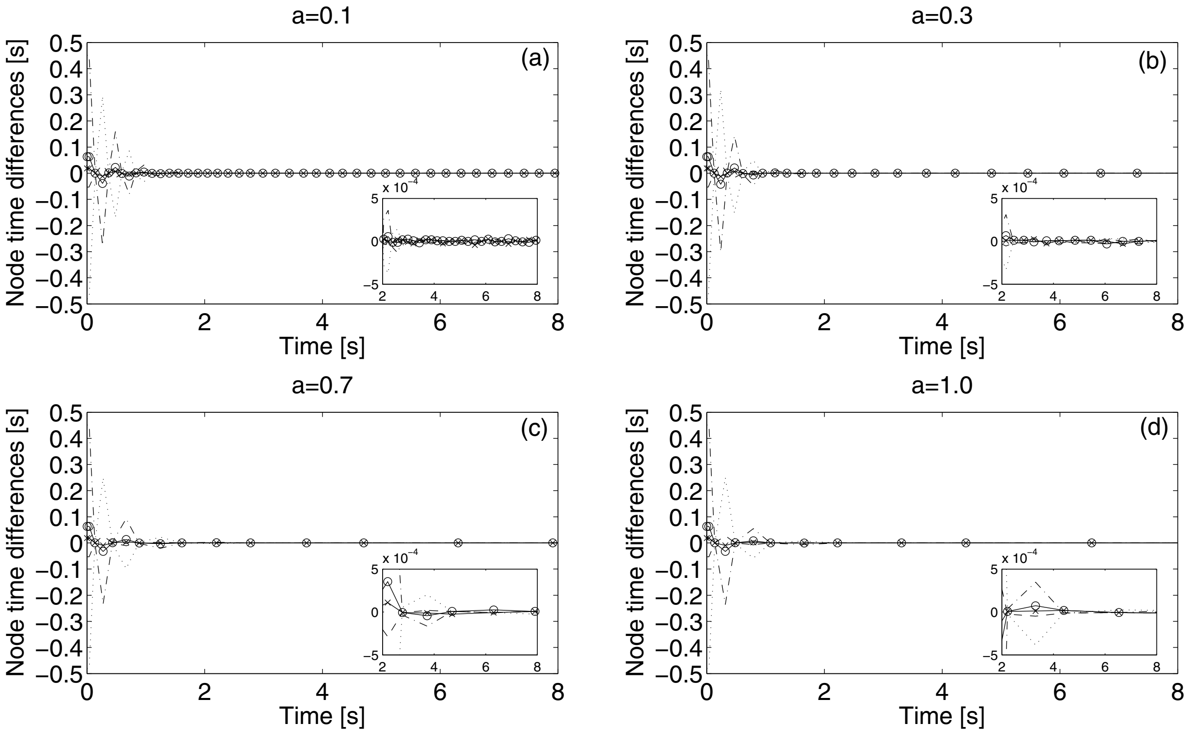


Fig. 2. Time differences between 5 nodes of a WSN for four values of the synchronization interval expansion coefficient a, i.e. $a = 0.1$ (a), $a = 0.3$ (b), $a = 0.7$ (c), $a = 1.0$ (d). The insets within each figure show a zoomed portion of the same patterns in the range $[3, 5]$ s.

for $i = 1, \dots, n$. Thus, the state of each timer x_i can be corrected by replacing the values of (16) into (3).

D. Synchronization Rescheduling

Immediately after correcting the timers, at first the vectors $\underline{\hat{\mathbf{x}}}_i$ must be reset for the reasons stated in Section III. Then, the next synchronization interval must be computed in order to schedule the following synchronization. To this purpose, the node

$$s = \begin{cases} \text{mod}(k+1, n) & \text{mod}(k+1, n) \neq 0 \\ n & \text{mod}(k+1, n) = 0 \end{cases} \tag{17}$$

which is supposed to become the next SM, adjusts the duration of the synchronization interval as follows:

$$\gamma_{k+1} = \begin{cases} \gamma_k(1+a) & \sigma^2_{x_k} < \varepsilon_k \\ \gamma_k & \sigma^2_{x_k} \geq \varepsilon_k \end{cases} \tag{18}$$

where $\sigma^2_{x_k}$ is the MSE of $\underline{\hat{\mathbf{x}}}_s$ before correcting the timers[2], ε_k is the MSE of the synchronization error resulting from the previous iteration and $a \geq 0$ is a design parameter. If $\sigma^2_{x_k} < \varepsilon_k$, then $\varepsilon_{k+1} = \sigma^2_{x_k}$. Conversely, if $\sigma^2_{x_k} \geq \varepsilon_k$, then $\varepsilon_{k+1} = \varepsilon_k$. In this way, the dwell–time requirements described in Section III are met. Observe that by using the proposed approach the duration of subsequent synchronization intervals may only increase. This is a major difference compared to the solution described in [6].

[2]The dependence on t introduced in Section III here has been removed for the sake of simplicity.

V. Simulation Results

The effectiveness of the proposed approach has been validated through several simulations in Matlab™. All simulations take advantage of the synchronization uncertainty models developed in previous research works (e.g., [16]). The simulated WSN consists of 5 nodes, whose timers are clocked by crystal oscillators (XO) running at $f_0 = 32768$ Hz, as in XBow TelosB™ or Tmote-Sky™ platforms. Nodes' timers are randomly initialized in the range $[0, 1]$ seconds. Clock relative systematic frequency skews are randomly chosen between ± 100 ppm, while the timing jitter caused by phase noise is in the order of 2 ns rms over 1 s. Communication latencies depend the offered network traffic and they are in the order of several ms [13]. The eigenvalues of the feedback matrices $\mathbf{K}^k$ are kept in the in the set $[0, \frac{4}{2+\alpha(\gamma_k-2)}]$ as explained in Section III. Fig. 1 shows the time values measured by the network nodes before (a) and after (b) running the synchronization algorithm for $a = 0$. The pattern of each node is univocally identified by a different line style. In Fig. 1(b) it is quite evident that all timers tend to converge to a common relative time–scale within a few seconds. Fig. 2 enables a clearer comparison of the performance of the algorithm in different conditions, namely for various values of the interval expansion coefficient a. Observe that in all cases the synchronization uncertainty tends to decrease. However, both the number of synchronizations and the convergence speed change considerably as a function of a. If the synchronization intervals increase by 10% or 30% [aggressive rescheduling – cases (a) and (b)], the differences between the time values of the nodes become in the order of ± 1 tick of the timer (i.e., ± 30.52 μs) after less than 3 seconds. Conversely, if the synchronization intervals increase by 70% or 100% [conservative rescheduling – cases (c) and (d)], the convergence speed is lower. In fact, the time differences between different nodes still oscillate between some tens and some hundreds of μs, before reaching ± 1 tick, which represents the lower bound for synchronization uncertainty. Certainly, the conservative approach is less computationally demanding because the number of synchronization events is much smaller. This is clearly shown in Fig. 3, where the duration of the intervals between two subsequent synchronizations is plotted as a function of time during 60 seconds of simulation. Notice that when $a = 0.7$ or $a = 1$, the synchronization intervals increase up to about 10 and 17 seconds, respectively. During long synchronization intervals the local clocks might occasionally diverge due to unexpected or time–varying phenomena, thus leading to synchronization uncertainty values larger than the theoretical lower bound. In order to reduce the probability of such events, the best value of a should result from the trade–off between the target synchronization accuracy required by a certain application and the computational burden and power consumption associated with frequent synchronizations.

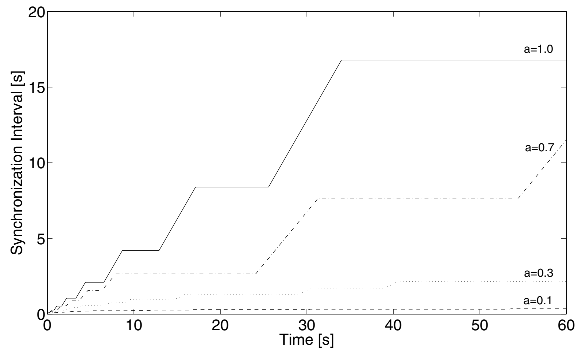


Fig. 3. Duration of the synchronization intervals for different values of the expansion coefficient a.

VI. Conclusion

In this paper, the problem of synchronizing the time measured by different WSN nodes is tackled by means of a distributed algorithm that drives the local timers towards a common timescale. The proposed solution represents an interesting attempt of combining together the effective time–stamping mechanism used in the Reference Broadcast System, an optimal clock correction technique based on control theory and a flexible-rate synchronization scheduling policy. The proposed solution is asymptotically stable. Indeed, the residual synchronization uncertainty can be reduced down to timer resolution, in spite of the random communication latencies affecting the synchronization procedure and the phase noise of the local oscillators. Various simulation results confirm that the convergence speed may be traded for the number of synchronization events. Further research activities are currently focused on robustness analysis in case of unstable radio links, in view of the implementation on commercial WSN platforms.

Acknowledgment

The results presented in this paper were obtained within the FP-7 EU project *Control of Heterogeneous Automation Systems* (CHAT), EC contract IST-2008-224428. Authors would like to thank the local coordinator of the project, Prof. Luigi Palopoli, for supporting this work.

References

[1] A. D. Angelis, M. Dionigi, A. Moschitta, R. Giglietti, and P. Carbone, "An experimental UWB distance measurement system," in *Proc. IEEE Instrumentation and Measurement Technology Conference (I2MTC)*, May 2008, pp. 1016–1020.

[2] *IEEE 1588:2008, Precision clock synchronization protocol for networked measurement and control systems*, New York, USA, July 2008.

[3] *IEEE 802.15.4:2006, Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - specific requirement. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, New York, USA, September 2006.

[4] S. Yoon, C. Veerarittiphan, and M. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks," *ACM Trans. on Sensor Networks (TOSN)*, vol. 3, no. 2, pp. 1–33, June 2007.

[5] S. Palchaudhuri, A. Saha, and D. Johnson, "Adaptive clock synchronization in sensor networks," in *Proc. of Int. Symposium on Information Processing in Sensor Networks (IPSN)*, Berkeley, California, USA, April 2004, pp. 340–348.

[6] A. Ageev, D. Macii, and A. Flammini, "Towards an adaptive synchronization policy for wireless sensor networks," in *Proc. of Int. Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, Ann Arbor, Michigan, USA, September 2008.

[7] S. Ganeriwal, R. Kumar, and Srivastava, "Timingsync protocol for sensor networks," in *Proc. of Int. Conf. on Embedded Networked Sensor Systems*, 2003, pp. 138–149.

[8] M. Maròti, B. Kusy, G. Simon, and A. Ldeczi, "The flooding time synchronization protocol," in *Proc. of Int. Conf. Embedded Networked Sensor Systems*, 2004, pp. 39–49.

[9] J. Elson, L. Girod, and D. Estrin, "Fine–grained network time syncronization using reference broadcasts," in *Proc. of Symposium on Operating Systems Design and Implementation*, 2002, pp. 147–163.

[10] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi–agent systems," *Proc. of IEEE*, vol. 95, no. 1, pp. 215–233, January 2007.

[11] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "A PI consensus controller for networked clocks synchronization," in *Proc. of 17th IFAC World Congress*, Seoul (Korea), July 2008.

[12] S. Bregni, "Clock stability characterization and measurement in telecommunications," *IEEE Trans. on Instrumentation and Measurement*, vol. 46, no. 6, pp. 1284–1294, December 1997.

[13] A. Ageev, D. Macii, and D. Petri, "Experimental characterization of communication latencies in wireless sensor networks," in *Proc. 16th Intl. Symp. Exploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurements (IMEKO TC4)*, Sesto Fiorentino, Italy, September 2008, pp. 258–263.

[14] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1069–1082, July 2000.

[15] D. Liberzon, *Switching in Systems and Control.* Birkhauser, January 2003.

[16] D. P. A. Ageev, D. Macii, "Synchronization uncertainty contributions in wireless sensor networks," in *Proc. of Int. Instrumentation and Measurement Technology Conference*, 2008, pp. 1986–1991.