Adaptive Control-Based Clock Synchronization in Wireless Sensor Networks

Kasım Sinan Yıldırım, *Member, IEEE,* Ruggero Carli, *Member, IEEE,* and Luca Schenato, *Member, IEEE*

***Abstract*— This paper presents PISync, a novel distributed synchronization algorithm based upon a Proportional-Integral (PI) controller for Wireless Sensor Networks (WSNs). PISync synchronizes each sensor node by applying a proportional feedback (P) and an integral feedback (I) on the relative synchronization error with respect to the received reference time which allow to simultaneously compensate both clock offset and frequency differences. We highlight the benefits of this approach in terms of improved steady state error and scalability as compared to least-squares based time synchronization, and we also propose an on-line adaptive strategy for the design of the integrator gain to further improve performance. We present practical flooding-based and fully-distributed protocol implementations of the PISync algorithm and show through real-world experiments that it has considerably better performance over FTSP, the de-facto time synchronization protocol in WSNs, in terms of both rate of convergence and steady-state error with the additional advantage of minimal resource requirement.**

I. Introduction

Wireless sensor networks (WSNs) perceive and manipulate physical world by means of their sensors and actuators, and interact with each other by means of wireless communication. Their clock systems are implemented using low-cost crystal oscillators that are intended to operate at specific nominal frequencies. Counter registers of these clock systems are clocked with these oscillators and they are increased with each pulse event, so called the tick of the clock. Clock drift occurs when the clock signal produced by the oscillator deviates from its intended nominal frequency which is a phenomenon occurring quite frequently mainly due to temperature changes. Hence, built-in clocks of the sensor nodes are not sufficient alone to provide synchronized time notion which is a fundamental requirement for collaboration and coordination of the sensor nodes [1]. Time synchronization is the process that establishes a common time notion among the sensor nodes in a WSN. This process should be adaptive since it requires coping with several environmental dynamics such as frequent temperature and topological changes, packet losses and quantization errors. Moreover, power, memory and computation constraints of the sensor nodes make time synchronization extremely challenging.

A network-wide time synchronization approach, which is quite simple and robust to dynamic topological changes in WSNs, is to disseminate the stable time information of one or more reference nodes through the network. This approach, referred as *flooding-based* time synchronization, allows each receiver node to calculate its relative offset and frequency differences with respect to the received time information, and to estimate the reference time. There are several successful implementations of flooding-based time synchronization that employs *least-squares regression*, as presented in [2], [3], [4], [5], [6], [7]. However, in least squares, the relative clock drifts and offsets among two or more nodes are estimated separately, thus giving rise to time-synchronization algorithms which are non-linear in the measurement noise. As a consequence, the effect of various error sources appears in the time synchronization error dynamics as *multiplicative* noise which makes the global time synchronization error to approximately grow *exponentially* with the diameter of the network, thus with poor performance scaling properties. Besides, least-squares has high overhead in terms of computation and memory since they require to store numerous measurements for each neighbouring node [8].

As an alternative to the method of least-squares, the method of maximum likelihood estimation [9], [10], belief propagation [11], and convex closure [12] have also been proposed. Nevertheless, these methods also have considerable computational and memory overheads and it is not clear whether they can be practically implemented in real WSNs since only simulative results are presented. Recently, a different approach based on control theory has been independently proposed in [13] and [14] where synchronization is achieved by using linear feedback on the measured local synchronization error. The major advantage of this approach is that the error sources appear as additive noise so that the global time synchronization error approximately grows as the square root of the network diameter.

A. Contributions

In this paper, inspired by [14], we consider time synchronization as a control problem and devise a new distributed synchronization algorithm, named *PISync*, which compensates the clock offsets and the differences in clock speeds based on a *Proportional-Integral (PI) controller*. In PISync, nodes achieve time synchronization by applying a proportional feedback (P) and an integral feedback (I) on measured relative offsets to compensate their clock offset and clock speed differences with respect to the received reference time. We provide a theoretical analysis to highlight the benefits of PISync in terms of improved steady state

This work is supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n. 257462 HYCON2 Network of excellence.

Kasım Sinan Yıldırım is with the Department of Computer Engineering, Ege University, 35100, İzmir, TURKEY (sinan.yildirim@ege.edu.tr).

Ruggero Carli and Luca Schenato are with the Department of Information Engineering.

error and scalability as compared to least-squares based time synchronization. Since PISync is linear, simple and easy to implement, it is perfectly suitable for WSNs. We introduce practical flooding-based and peer-to-peer protocol implementations of the PISync algorithm and an adaptive strategy for the integral gain in order to balance the trade-off between the convergence time and the steady state error in dynamic environments. We present an experimental comparison of PISync with Flooding Time Synchronization Protocol (FTSP) [2], the de facto least-squares based time synchronization protocol in WSNs, on a testbed of 20 MICAz sensor nodes. At the light of our theoretical findings and experiments, the main advantages of the PISync over least-squares based time synchronization can be summarized as follows: (i) it is scalable in terms of steady state global synchronization error, which grows with the square root of the network diameter, (ii) it does not store distinct time information and it allocates only one third of RAM space as compared to least-squares based time synchronization, (iii) it is lightweight in terms of CPU usage since it requires more than an order of magnitude fewer operations as compared to the least-squares based time synchronization.

II. System Model

In this section, we propose a hardware clock model, a logical clock model and a network model which we use for the presentation and analysis of the synchronization algorithms in this article.

A. Hardware and Logical Clock Model

Assume each hardware clock has an oscillator capable to produce an event at time $t(k)$, $k \in \mathbb{N}$ and let $s(t)$ be the counter of these events, namely, $s(t) = \sum_{k=0}^{\infty} \mathbf{1}(t - t(k))$ where $\mathbf{1}$ is the unit step function. In this way the counter output is the step shaped function shown in Figure 1. Notice that the function

$$f(t) := \frac{1}{t(k+1) - t(k)} \qquad \forall t \in [t(k), t(k+1)[, \quad (1)$$

can be interpreted as the oscillator frequency at time t and that $s(t) \simeq \int_{-\infty}^{t} f(\sigma)d\sigma$. Typically a nominal value $\hat{f}$ of $f(t)$ is known together with a lower bound f_{min} and an upper bound f_{max} such that $f(t) \in [f_{min}, f_{max}] = [\hat{f} - \Delta f_{max}, \hat{f} + \Delta f_{max}]$ where $\Delta f_{max} = \frac{f_{max} - f_{min}}{2}$. From the counter $s(t)$, one can build a time estimate $\hat{t}(t)$ by letting

$$\hat{t}(t) = \hat{t}(t_0) + \hat{\Delta}(t)[s(t) - s(t_0)], \quad (2)$$

where $\hat{\Delta}(t)$ is an estimate of the oscillation period $1/f(t)$ in the period $[t_0, t]$. It is reasonable to initialize $\hat{\Delta}(t)$ to $1/\hat{f}$. The time estimate $\hat{t}(t)$ can be considered as the value of the **logical clock** at time t and represents the network-wide global time. It can be observed that the estimate $\hat{\Delta}(t)$, which we will also refer to as the **rate multiplier**, represents the progress rate (speed) of the logical clock.

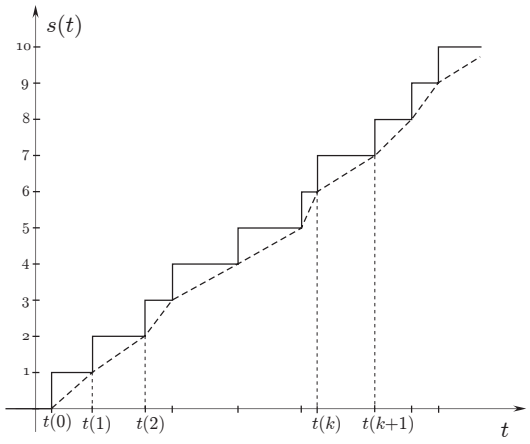

Fig. 1: The graphs of $s(t)$ (continuous line) and of its approximation $\int_{-\infty}^{t} f(\sigma)d\sigma$ (dashed line). Due to their dynamic frequencies, the tick events of clocks do not occur at regular time intervals. Hence, $s(t)$ is a step shaped function.

B. Network Model

In our setup, we represent a communication network by a graph $G = (V, \mathcal{E})$ where the vertex set $V = \{1, \ldots, N\}$ represents sensor nodes. We assume that if $(i, j) \in \mathcal{E}$ in graph G then node i can communicate to the node j. Specifically, each node $i \in \{1, \ldots, N\}$ broadcasts $\hat{t}_i(t)$ to its neighbours at time instants $T_{\mathrm{tx},i}(h)$, $h = 0, 1, \ldots$, and can use any information it receives from the neighbouring nodes to apply a **control** at the time instants $T_{\mathrm{up},i}(h)$, $h = 0, 1, \ldots$. More precisely, node i can modify both $\hat{t}_i(t)$ and $\hat{\Delta}_i(t)$ whenever it obtains information allowing it to improve its time and oscillator frequency estimates. In this paper we consider additive corrections of the form [1]

$$\begin{aligned} \hat{t}_i(T^+_{\mathrm{up},i}(h)) &= \hat{t}_i(T_{\mathrm{up},i}(h)) + u'_i(h) \\ \hat{\Delta}_i(T^+_{\mathrm{up},i}(h)) &= \hat{\Delta}_i(T_{\mathrm{up},i}(h)) + u''_i(h) \end{aligned} \quad (3)$$

where $u'_i(h)$ and $u''_i(h)$ denote the **control inputs** applied to $\hat{t}_i$ and $\hat{\Delta}_i$, respectively, at time $T_{\mathrm{up},i}(h)$. Moreover, for $t \in \left[T^+_{\mathrm{up},i}(h), T_{\mathrm{up},i}(h+1)\right]$ we assume that $\hat{t}_i(t)$ is updated according to (2), while $\hat{\Delta}_i$ is left unchanged, i.e.:

$$\begin{aligned} \hat{t}_i(t) &= \hat{t}_i(T^+_{\mathrm{up,i}}(h)) + \hat{\Delta}_i(T^+_{\mathrm{up,i}}(h))\left(s_i(t) - s_i(T_{\mathrm{up},i}(h))\right), \\ \hat{\Delta}_i(t) &= \hat{\Delta}_i(T^+_{\mathrm{up},i}(h)) \end{aligned} \quad (4)$$

In Figure 2 we depict the behaviour of the logical clock $\hat{t}_i$ and of the rate multiplier $\hat{\Delta}_i$.

The objective is to find a control strategy leading the logical clocks $\hat{t}_i(t)$, $i \in \{1, \ldots, N\}$ to obtain the same time estimate, namely, such that there exist constants $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}$ such that synchronization errors

$$e_i(t) := \hat{t}_i(t) - (at + b), \quad i \in \{1, \ldots, N\}, \quad (5)$$

converge to zero or remain small.

[1] Given time t, with the symbol t^+ we mean the time instant just after t.

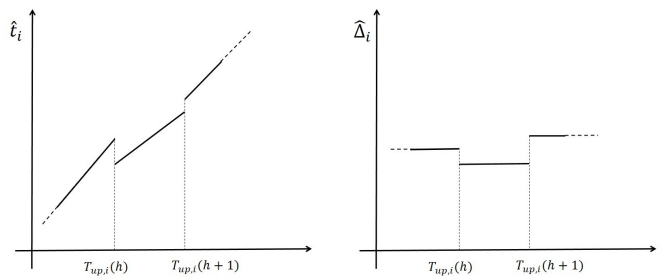

Fig. 2: Behavior of the logical clock $\hat{t}_i$ (left column) and the rate multiplier $\hat{\Delta}_i$ (right column), respectively.

III. PISYNC ALGORITHM

In this section, we devise a distributed synchronization algorithm which is based upon a Proportional-Integral (PI) Controller. Our algorithm, named *PISync*, applies a proportional feedback (P) and an integral feedback (I) on measured relative offsets to compensate the differences between the clock offsets and the clock speeds, respectively. For ease of presentation, consider the pairwise synchronization between two nodes i and j, where i plays the role of the reference clock. Without loss of generality, assume that node i transmits, at a generic transmission time $T_{\mathrm{tx},i}$, the information $\hat{t}_i(T_{\mathrm{tx},i})$ to node j. Due to transmission delays, the information $\hat{t}_i(T_{\mathrm{tx},i})$ is received by node j at a delayed time $T_{\mathrm{tx},i}+\gamma_{i,j}$ where $\gamma_{i,j}$ is a non-negative real number representing the deliver delay between i and j. Based on the information received, node j instantaneously applies the following PISync update rule

$$\begin{aligned} u'_j &= \beta_j \left(\hat{t}_i(T_{\mathrm{tx},i}) - \hat{t}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right)\right) \\ u''_j &= \alpha_j \left(\hat{t}_i(T_{\mathrm{tx},i}) - \hat{t}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right)\right) \end{aligned} \tag{6}$$

and corrects itslogical clock $\hat{t}_j(T_{\mathrm{tx},i}+\gamma_{i,j})$ and rate multiplier $\hat{\Delta}_j(T_{\mathrm{tx},i}+\gamma_{i,j})$, where β_j, α_j are two control parameters to be designed. Observe that the update rule above requires only the *estimated synchronization error* between nodes i and j, and the two control parameters α_j, β_j to be designed. Using PISync update rule, we can rewrite (3) as follows since we have that $T_{\mathrm{up},j} = T_{\mathrm{tx},i}+\gamma_{i,j}$:

$$\begin{aligned} \hat{t}_j\left((T_{\mathrm{tx},i}+\gamma_{i,j})^+\right) &= \hat{t}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right) \\ &\quad + \beta_j\left(\hat{t}_i(T_{\mathrm{tx},i}) - \hat{t}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right)\right) \\ \hat{\Delta}_j\left((T_{\mathrm{tx},i}+\gamma_{i,j})^+\right) &= \hat{\Delta}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right) \\ &\quad + \alpha_j\big(\hat{t}_i(T_{\mathrm{tx},i}) - \hat{t}_j\left(T_{\mathrm{tx},i}+\gamma_{i,j}\right)\big) \end{aligned} \tag{7}$$

Additionally in realistic scenarios, the information received by node j is affected by a noise $v_{i,j}$, which models the unavoidable quantization effects and communication channels errors. Hence, in the above equations the term $\hat{t}_i(T_{\mathrm{tx},i})$ should be replaced by $\hat{t}_i(T_{\mathrm{tx},i}(h)) + v_{i,j}(T_{\mathrm{tx},i}(h))$.

A. Synchronization Conditions and Rate of Convergence

A preliminary attempt for the design of the parameters α_j and β_j can be obtained under some explicit assumptions that allows for an explicit mathematical analysis of the convergence rate and the steady-state error of the PISync algorithm. Assume that the oscillator frequencies f_i, f_j are constant, i.e. $f_i(t) = \bar{f}_i, \quad f_j(t) = \bar{f}_j$ for all $t \in \mathbb{R}_{>0}$, node i is perfectly synchronized with respect to the absolute time, i.e. $\bar{f}_i = \hat{f}$ and $\hat{t}_i(0) = 0$, and node i periodically transmits to node j a message carrying its own logical clock with period B. In addition assume that only node j updates its logical clock by keeping $\beta_j(t)$ and $\alpha_j(t)$ constant, i.e., $\beta_j(t) = \beta$ and $\alpha_j(t) = \alpha$. Then, we have the following result.

Proposition III.1 *Under the assumption of no communication delay and no transmission error, i.e. $\gamma_{i,j} = 0$ and $v_{i,j} = 0$, and of periodic communication with period B, synchronization is achieved if and only if*

$$0 < \beta < 2, \quad 0 < \alpha < \frac{2(2-\beta)}{\bar{f}_j B} \tag{8}$$

holds. Moreover, for $\beta = 1$, the convergence rate factor ρ (such that $|\hat{t}_i(t) - \hat{t}_j(t)| \propto \rho^{\frac{t}{B}}$) is given by

$$\rho = \left|1 - \alpha \bar{f}_j B\right|, \tag{9}$$

and it is minimised for

$$\alpha^* = \frac{1}{\bar{f}_j B} \tag{10}$$

Proof: Since node i is perfectly synchronized with respect to the absolute time, Eqn.(5) becomes

$$e_j(t) = \hat{t}_j(t) - \hat{t}_i(t) = \hat{t}_j(t) - \left(\frac{\bar{f}_i}{\hat{f}} t + \hat{t}_i(0)\right) = \hat{t}_j(t) - t. \tag{11}$$

Moreover, since $\gamma_{ji} = 0$, we have $T_{tx,i}(h) = hB, \quad h \in \mathbb{N}$. Let us denote $\hat{t}_j(h) = \hat{t}_j(hB)$ and similarly for $\hat{t}_i$ and e_j, therefore the update in Eqn.(7) becomes

$$\hat{t}_j(h^+) = \hat{t}_j(h) - \beta_j e_j(h) \tag{12}$$

$$\hat{\Delta}_j(h^+) = \hat{\Delta}_j(h) - \alpha_j e_j(h). \tag{13}$$

By considering Eqn. 11 for $t = (h+1)B$ and by recalling the definition of e_j and that $\hat{t}_i(h+1) = \hat{t}_i(h) + B$, we obtain

$$\begin{bmatrix} e_j(h+1) \\ \hat{\Delta}_j(h+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1-\beta-\alpha B\bar{f}_j & B\bar{f}_j \\ -\alpha & 1 \end{bmatrix}}_{F} \begin{bmatrix} e_j(h) \\ \hat{\Delta}_j(h) \end{bmatrix} - \begin{bmatrix} B \\ 0 \end{bmatrix} \tag{14}$$

where $\begin{bmatrix} e_j(0) \\ \hat{\Delta}_j(0) \end{bmatrix} = \begin{bmatrix} \hat{t}_j(0) \\ \frac{1}{\hat{f}} \end{bmatrix}$ being $\bar{f}_j$, $\hat{t}_j(0)$ arbitrary.

By inspecting these dynamics, it is immediate to see that, if the gains α and β are chosen such that the matrix F is **strictly stable**, i.e. its two eigenvalues have modulus strictly smaller than unity, then the dynamical system must converge asymptotically to a steady state, i.e. $\lim_{h\to\infty} e_j(h) = e_j(\infty)$ and $\lim_{h\to\infty} \hat{\Delta}_j(h) = \hat{\Delta}_j(\infty)$ which must satisfy

$$\hat{\Delta}_j(\infty) = \hat{\Delta}_j(\infty) - \alpha e_j(\infty) \Longrightarrow e_j(\infty) = 0, \quad (\alpha \neq 0)$$
$$e_j(\infty) = (1-\beta-\alpha B\bar{f}_j) e_j(\infty) + B\bar{f}_j \hat{\Delta}_j(\infty) - B \Longrightarrow \hat{\Delta}_j(\infty) = \tfrac{1}{\bar{f}_j}$$

which shows that time synchronization is eventually achieved, since $e_j(\infty) = 0$. The eigenvalues of the matrix

F which are given by the solution of the following second order system

$$z^2 - (2 - \beta - \alpha B \bar{f}_j) z + (1 - \beta) = 0$$

can be calculated as

$$z_{1,2} = \frac{2 - \beta - \alpha B \bar{f}_j \pm \sqrt{(\beta + \alpha B \bar{f}_j)^2 - 4\alpha \bar{f}_j B}}{2}.$$

We then would like to find the set of values for the gains α and β for which these roots are stable, i.e., $|z_{1,2}| < 1$. After some simple calculations, we obtain:

$$0 < \beta < 2, \quad 0 < \alpha < \frac{2(2-\beta)}{\bar{f}_j B}. \tag{15}$$

It is also possible to find the optimal value for α for any fixed and feasible value of β to maximize the rate of convergence, which after some simple calculations is given by

$$\alpha^* = \text{argmin}_\alpha |z_{1,2}| = \frac{2-\beta}{\bar{f}_j B} \tag{16}$$

When $\beta = 1$ one can easy see that $z_1 = 0, z_2 = 1 - \alpha \hat{f} B$. This concludes the proof. ■

Remark III.2 Observe from (14) that the dynamics of $\hat{\Delta}_j$ is a discrete time integrator driven by the error signal e_j where the parameter α is referred as the *integrator feedback gain*. The dynamics of the synchronization error e_j are affected by the output of the integrator $\hat{\Delta}_j$ and includes also a proportional feedback on itself via the *proportional feedback gain* β. The role of the proportional gain is to compensate for the different initial clock offset $\hat{t}_j(0)$, while the role of the integrator gain is to compensate for different clock drifts $\bar{f}_j - \hat{f}$, which are both not directly measurable. In fact, if the integrator is disabled and the proportional gain is chosen to keep the dynamics stable, then the a steady state error is present:

$$\alpha = 0 \Longrightarrow \begin{cases} \hat{\Delta}_j(h) = \frac{1}{\hat{f}}, \; e_j(h+1) = (1-\beta) e_j(h) + B \frac{\bar{f}_j - \hat{f}}{\hat{f}} \\ e_j(\infty) = \frac{B(\bar{f}_j - \hat{f})}{\beta \hat{f}}, \quad (0 < \beta < 2) \end{cases}$$

Note that the steady state error is directly proportional to both the difference of the relative clock speed and the transmission period B. In fact, if all clocks have the same drift, i.e. $\bar{f}_j = \hat{f}, \forall j$, then the proportional feedback alone would suffice to drive the synchronization error to zero.

B. Steady State Error

After presenting the sufficient conditions to establish synchronization and the convergence rate of PISync, we now focus its steady-state error by considering a more realistic scenario including transmission errors and time-varying clock frequencies. Specifically we assume that

- quantization effects and communication channels' errors are modeled as zero mean white noise of variance $\eta_t^2 \frac{1}{\hat{f}^2}$ where η_t is an adimensional parameter which is typically in the order of unity;
- for $t \in [hB, (h+1)B]$, the frequency $\bar{f}_k(t)$, $k = i, j$, is given by $\bar{f}_k + w_k(h)$ where $w_j(h)$ is a zero mean-noise uniformly distributed in $[-\Delta f_{max}, \Delta f_{max}]$ with corresponding variance $\eta_w^2 \hat{f}^2$ where η_w is an adimensional parameter which refers to the typical relative frequency change over one synchronization period B.

Proposition III.3 *Assume the transmission errors and the frequencies $\bar{f}_k(t)$, $k = i, j$, are modeled as above. Under the assumption of no communication delay and of periodic communication with period B, for $\beta = 1$, the root-mean-square error (RMSE) σ_{RMSE} of node j is a monotonically increasing function on α and is given by*

$$\sigma_{RMSE} = \sqrt{\frac{2\alpha B^2 \left(\eta_t^2 + \eta_w^2 \hat{f}^2 B^2\right)(1+\eta_w^2)}{2B\hat{f} - \alpha B^2 \hat{f}^2 (1+\eta_w^2)} + \eta_t^2 \frac{1}{\hat{f}^2} + 2\eta_w^2 B^2} \tag{17}$$

Proof: Recall that $T_{\text{tx},i}(h) = hB$, where B is a given sampling time and $h = 0, 1, 2, \ldots$. Since node i is the reference node we get that $\hat{\Delta}_i(h) = 1/\hat{f}$ for all h. Hence

$$\hat{t}_i(h+1) \;=\; \hat{t}_i(h) + \frac{1}{\hat{f}} (\hat{f} + w_i(h)) B$$

and

$$\begin{aligned} \hat{t}_j(h+1) &= \hat{t}_j(h^+) + \hat{\Delta}_j(h^+) (\hat{f} + w_j(h)) B \\ \hat{\Delta}_j(h+1) &= \hat{\Delta}_j(h^+) \end{aligned}$$

where

$$\hat{t}_j(h^+) = \hat{t}_j(h) - \beta(e_j(h) - v_{i,j}(h)) \tag{18}$$

$$\hat{\Delta}_j(h^+) = \hat{\Delta}_j(h) - \alpha(e_j(h) - v_{i,j}(h)) \tag{19}$$

being $v_{i,j}(h)$ the transmission error. Recalling that $e_j(h) = \hat{t}_j(h) - \hat{t}_i(h)$, and by defining $z_j(h) = \hat{\Delta}_j(h)\hat{f} - 1$, then after some straightforward calculation we get:

$$\begin{aligned} e_j(h+1) = v_{i,j}(h) + B \left(1 + w_j(h) \frac{1}{\hat{f}}\right) z_j(h+1) \\ + \frac{B}{\hat{f}} (w_j(h) - w_i(h)) \end{aligned}$$

where $z_j(h+1) = z_j(h) - \alpha \hat{f}(e_j(h) - v_{i,j}(h))$. By substituting $e_j(h)$ in this equation above, we get:

$$\begin{aligned} z_j(h+1) = \left(1 - \alpha \hat{f} B - \alpha w_j(h-1) B\right) z_j(h) \\ - \alpha \hat{f} \left(v_{i,j}(h-1) + \frac{B}{\hat{f}} (w_j(h-1) - w_i(h-1)) - v_{i,j}(h)\right). \end{aligned}$$

Let $P_{z_j}(h) = \mathbb{E}\left[z_j^2(h)\right]$ and $P_{e_j}(h) = \mathbb{E}\left[e_j^2(h)\right]$. It follows

$$\begin{aligned} P_{z_j}(h+1) = \left((1 - \alpha \hat{f} B)^2 + \alpha^2 \sigma_w^2 B^2\right) P_{z_j}(h) \\ + \alpha^2 \left(2\sigma_v^2 \hat{f}^2 + 2\sigma_w^2 B^2\right) \end{aligned}$$

Hence

$$\lim_{h \to \infty} P_{z_j}(h) = \frac{2\alpha \left(\eta_t^2 + \eta_w^2 \hat{f}^2 B^2\right)}{2B\hat{f} - \alpha B^2 \hat{f}^2 (1 + \eta_w^2)}$$

and, in turn,

$$\bar{P}_{e_j} = \lim_{h\to\infty} P_{e_j}(h)$$
$$= \frac{2\alpha B^2\Big(\eta_t^2 + \eta_w^2 \hat{f}^2 B^2\Big)\big(1+\eta_w^2\big)}{2B\hat{f} - \alpha B^2 \hat{f}^2 (1+\eta_w^2)} + \eta_t^2 \frac{1}{\hat{f}^2} + 2\eta_w^2 B^2.$$

This concludes the proof. ∎

Remark III.4 Notice, from expression in (17), that, due to the presence of noises, even for $\alpha = 0$ there is still some non-zero steady state error. However, since the expression in (17) is increasing on α, the smaller the value of α is, the smaller the value of the error is, even though, as α approaches 0, the algorithm becomes slower and slower.

C. Comparison with Least-squares Based Synchronization

Under the same hypothesis of the previous two sections, we now provide a similar intuitive presentation of the **least-squares-based time synchronization**. In this context, the logical clocks of the two nodes can be written as:

$$\hat{t}_i(h) = \frac{1}{\hat{f}} s_i(h)$$
$$\hat{t}_j(h) = u_j'(h) + u_j''(h) s_j(h)$$

where $u_j'(h)$ and $u_j''(h)$ have to be designed to drive the synchronization error $e_j(h) = e_j(h) - e_i(h)$ to zero. We assume that $u_j'(h) = u'$ and $u_j''(h) = u''$ are kept constant for the first H steps so that we can write

$$\underbrace{\begin{bmatrix} e_j(H-1) \\ \vdots \\ e_j(0) \end{bmatrix}}_{\mathbf{e}} = \underbrace{\begin{bmatrix} 1 & s_j(H-1) \\ \vdots & \vdots \\ 1 & s_j(0) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u' \\ u'' \end{bmatrix}}_{\mathbf{u}} - \underbrace{\frac{1}{\hat{f}} \begin{bmatrix} s_i(H-1) \\ \vdots \\ s_i(0) \end{bmatrix}}_{\mathbf{b}}$$

Under least-square-based synchronization the values for the compensating parameters is given by

$$\text{argmin}_{\mathbf{u}} \|\mathbf{e}\| = \text{argmin}_{\mathbf{u}} \|A\mathbf{u} - \mathbf{b}\| \Longrightarrow \mathbf{u} = (A^T A)^{-1} A^T \mathbf{b}$$

Note that A and b are known to node j as long as node i transmits either $\hat{t}_i(h)$ or $s_i(t)$ since $\hat{f}$ is known. In the specific case when $H = 2$, the solution is given by:

$$u' = \frac{1}{\hat{f}} \frac{s_i(0)s_j(1) - s_i(1)s_j(0)}{s_j(1) - s_j(0)}, u'' = \frac{1}{\hat{f}} \frac{s_i(1) - s_i(0)}{s_j(1) - s_j(0)}.$$

If no measurements errors are considered, then the previous solutions provide exact synchronization, i.e., $\hat{t}_j(h) = \hat{t}_i(h), \forall h \geq 2$. In practice, this is not the case due, for example, to transmission delay or quantization, and therefore the previous procedure has to be repeated periodically.

At the light of the derivations above about the control-based and least-squares-based time synchronization a number of observations are in order: (i) Under the ideal scenarios above, the rate of convergence for the control-based synchronization is asymptotic, while using least-squares can be achieved in finite time. (ii) If measurements errors are present, due for example to transmission delay or quantization, then we have to substitute $\hat{t}_i(h) \leftarrow \hat{t}_i(h) + w_{i,j}(h)$ into Eqn. (14) and $s_i(h) \leftarrow s_i(h) + v_{i,j}(h)$ into Eqn. (III-C), where $v_{i,j}(h)$'s represent the measurement noise at iteration h. As the consequence the time synchronization error dynamics for the control-based strategy becomes:

$$\begin{bmatrix} e_j(h+1) \\ \hat{\Delta}_j(h+1) \end{bmatrix} = \begin{bmatrix} 1-\beta_j & B\bar{f}_j \\ -\alpha_j & 1 \end{bmatrix} \begin{bmatrix} e_j(h) \\ \hat{\Delta}_j(h) \end{bmatrix} - \begin{bmatrix} B \\ 0 \end{bmatrix} + v_{i,j}(h) \begin{bmatrix} \beta_j \\ \alpha_j \end{bmatrix} \quad (20)$$

while the equation u'' for the first iteration in the least-squares approach becomes

$$u'' = \frac{1}{\hat{f}} \frac{s_i(1) - s_i(0) + w_{i,j}(1) - w_{i,j}(0)}{s_j(1) - s_j(0) + w_{i,j}(1) - w_{i,j}(0)}$$

These two equations clearly show that in control-based approach the disturbances enter the synchronization error dynamics **linearly**, while in the least-square dynamics **non-linearly**. Also note that in the control-based approach the measurement errors are amplified by the control gains α_j, β_j, which therefore should be kept small to reduce the steady state error. However, this comes at the price of slower convergence rate since the modulus of the largest eigenvalue of F approaches unity for $\alpha_j \to 0, \beta_j \to 0$. (iii) Memory and CPU requirements for the control-based strategy are minimal since only two additions and two multiplication are needed and no data storage is necessary. Differently, in the least-squares-based approach it is necessary to store $2H$ measurements and perform $3H$ multiplications, $3H$ additions and a 2×2 matrix inversion and multiplication.

IV. Multi-Hop Time Synchronization with PISync

Employing PISync algorithm, network-wide time synchronization can efficiently be established by disseminating the reference time through the network via *flooding*. To this end, a special reference node which is predefined or dynamically elected is required to broadcast its stable time information into the network periodically. The nodes that are in the communication range of the reference node can directly receive the time information of the reference node and synchronize their logical clocks according to PISync algorithm. In order to inform their neighboring nodes about the reference time, these nodes are also required to broadcast the value of their logical clocks periodically so that their neighbors receive the estimated reference time and employ the PISync algorithm to synchronize their logical clocks. As discussed previously, this multi-hop synchronization approach is lightweight in terms of memory and computation requirements as compared to least-squares based synchronization. Moreover, as a direct application of Proposition III.3, one can show that the root mean synchronization error of this approach grows with the *square root* of the distance to the reference node.

In the previous section, we considered an idealized pairwise synchronization scenario and obtained the constant value α^* of the integral gain α that will give rise to the maximum rate of convergence for the fixed proportional gain

$\beta = 1$, as shown in Eqn. (8). However, one cannot get the minimum steady-state error and fastest synchronization with these constant gains. In fact, shown in Eqn. (17), for $\beta = 1$ the optimal value of α to minimize the steady state error is $\alpha = 0$. Hence, in this section we introduce a strategy to adjust control gain α adaptively to improve both convergence rate and stability, especially for multi-hop time synchronization.

We propose to use the constant value of $\beta = 1$ for the proportional gain since it is convenient to compensate the offset between the reference clock and the receiver clock in one step. Regarding the integral gain, since it compensates frequency differences of the clocks, its absence would give rise to the typical saw-tooth behaviour of the time synchronization error. However, if the integral gain is too large, it might drive the algorithm to instability. Besides, if the integrator is active when the measured synchronization is not due to the different clock speeds, the so-called windup problem occurs. Considering these issues, we propose the following procedure for the on-line adaptation of the integrator gain α. First we define the quantity e_{max} as follows:

$$e_{max} = \max_{i,j} \frac{|\bar{f}_i - \bar{f}_j|}{\hat{f}} B = \frac{2\Delta f_{max}}{\hat{f}} B \qquad (21)$$

which represents the maximum synchronization error that can be observed due to the frequency difference between two arbitrary nodes within a time window of size B. If any node observes that its synchronization error with respect to the reference time is greater than e_{max}, then it disables the integrator to avoid the integrator windup, since it is likely that the error is due to the large initial clock offset rather than the different clock frequency. After the proportional feedback compensates the initial clock offset differences, the observed synchronization error will be smaller than e_{max} and thus the integrator feedback will be enabled by setting the integrator gain to α^* that gives the fastest convergence rate. Since this value of the integrator gain is not sufficient to achieve the smallest steady-state error, we gradually reduce it at steady-state to decrease the synchronization error. In case of a sudden clock frequency change, the integrator gain is increased to quickly compensate frequency differences and after the algorithm has converged to steady-state the integrator gain is decreased again to reduce steady-state error. Following this simple intuition, we formalize our integrator gain adaptation of any node j where node i is the reference node with the following algorithm:

Algorithm 1 Adaptation algorithm for the integrator gain.

1: **if** $|e_{ij}(h)| > e_{max}$
2: $\quad \alpha_j(h) = 0$
3: **else if** $\alpha_j(h-1) = 0$
4: $\quad \alpha_j(h) = \alpha^*$
5: **else if** $\delta e_{ji}(h)\delta e_{ji}(h-1) > 0$
6: $\quad \alpha_j(h) = \max\{\lambda^+ \alpha_j(h-1), \alpha^*\}$
7: **else**
8: $\quad \alpha_j(h) = \frac{\alpha_j(h-1)}{\lambda^-}$

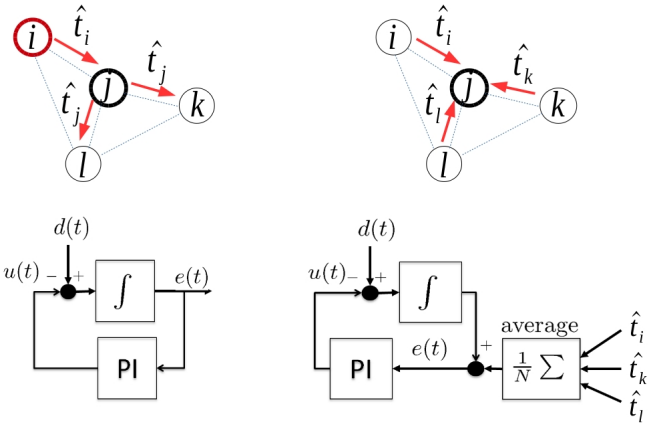

Fig. 3: Flooding-based (left) and fully-distributed (right) time synchronization approaches with PISync from node j's view in a network of 4 nodes.

Here, $\delta e_{ji}(h) = e_{ji}(h) - e_{ji}(h-1)$ initialized to $\delta e_{ji}(0) = \delta e_{ji}(-1) = 0$. The intuition behind the procedure is that if the variation of the error of the previous round $\delta e_{ji}(h-1)$ have the same sign then the integrator gain can be increased to accelerate the decrease of the error and hence the convergence rate. Conversely, if the signs of the error variation are opposite, then the error is oscillating and in order to decrease the steady-state error it is necessary to decrease the integral gain. The values $\lambda^+ = 2$ and $\lambda^- = 3$ are obtained empirically based on experimental performance analysis provided in [15].

A. Fully Distributed Network-wide Synchronization

PISync algorithm can also be applied to scenarios where nodes interact only with their direct neighbours in a peer-to-peer fashion and there is no any special node that floods the reference time information. In such fully-distributed executions, flooding-based synchronization approach can be slightly modified to allow nodes to synchronize to their direct neighbours. For this purpose, nodes are required to broadcast their clock values periodically to inform their neighbouring nodes about their current time information. Upon a new packet reception, each neighboring node calculates its synchronization error with respect to the received clock value, adds this error to its sum variable and increments the number of received clock values by one. It should be noted that this is a completely blind operation since there is no need to know the sender node or to store its clock value. At specific time instants, e.g. before broadcasting its clock value, each node calculates the average synchronization error inside its neighborhood by dividing its sum variable by the number of received clock values, and applies the PISync algorithm considering the average error to update its clock.

Figure 3 presents a comparison of flooding-based and fully-distributed time synchronization with PISync from the perspective of node j on a network of 4 nodes. Due to the frequency differences of the clocks, a constant disturbance

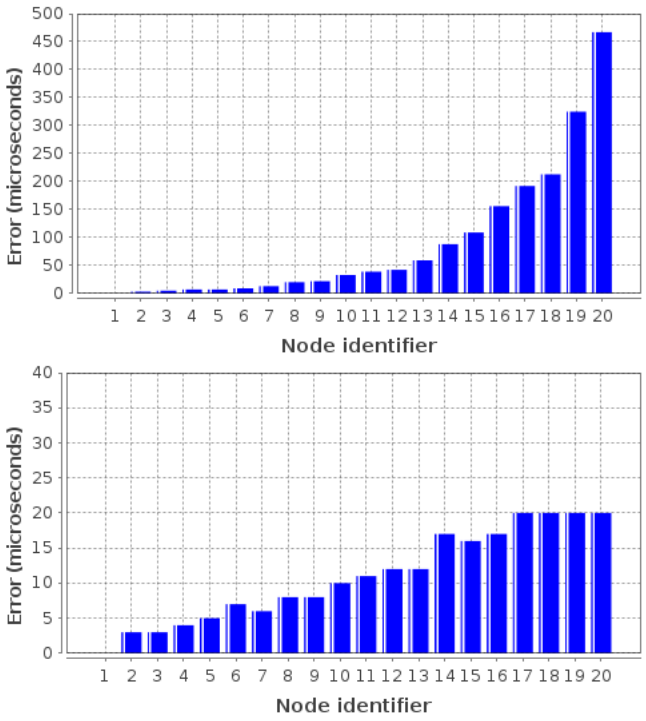

Fig. 4: Maximum error to the reference node on the line topology of 20 sensor nodes for FTSP (top) and flooding-based PISync (bottom), respectively.

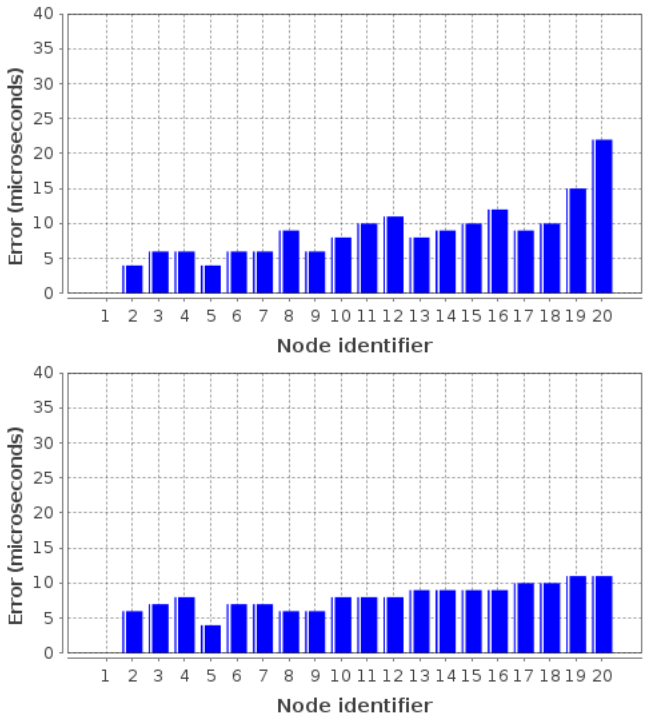

Fig. 5: Maximum error to the node with identifier 1 on the 5x4 grid topology for FTSP (top) and fully-distributed PISync (bottom), respectively.

$d(t)$ enters the system. In flooding-based approach, since node i is the reference node that floods its time information $\hat{t}_i(t)$, node j only considers its synchronization error $e(t) = \hat{t}_j(t) - \hat{t}_i(t)$ with respect to the reference time to apply the input $u(t) = -k_P e(t) + k_I \int e(t)dt$. On the other hand, in fully-distributed scenario, node j considers all clock values of its neighboring nodes to calculate the average synchronization error, i.e. $e(t) = 1/N \sum_{i,j} \hat{t}_j(t) - \hat{t}_i(t)$.

V. Evaluation

In this section, we present an experimental evaluation of PISync on a real testbed of 20 MICAz sensor nodes. For performance comparison, we choosed FTSP which is the de facto flooding based time synchronization protocol in WSNs and it employs least-squares regression. We performed the experiments on the line topology in order to observe the scalability of the flooding-based scenarios. In addition, we performed experiments on the 5x4 grid topology to obtain an impression about the performance of fully-distributed approaches. We set the beacon period B to 30 seconds and fixed the capacity of the least-squares regression table of FTSP to 8 elements. We assigned unique identifiers from 1 to 20 for each node and fixed node 1 as the reference node that floods the value of its clock through the network. Since the nominal frequency of the MICAz sensor nodes is $\hat{f} = 1$ MHz and their reported the nominal drift is ± 100 ppm, we substituted these values into Eqn.(21) and Eqn.(9) and used the following parameter values $e_{max} = 0.006$ and $\alpha^* = 1/(\hat{f}B) = 3,33 \cdot 10^{-8}$ in our experiments.

During the experiments, we collected instantaneous logical clock values from the nodes and calculated the maximum instantaneous synchronization error to the reference node. Figure 4 presents the summary of these calculations on the line topology for flooding-based PISync and FTSP, respectively. It can be observed that the synchronization error of FTSP is exponential with the distance to the reference node 1. Even in small network, we observed more or less 0.5 milliseconds synchronization error to the reference node. On the other hand, although PISync and FTSP have the same communication pattern, PISync outperformed FTSP drastically since we observed a synchronization error of at most 20 microseconds, which is a significant improvement as compared to FTSP. Moreover, the synchronization error grows linearly unlike the exponential growth of FTSP, thus confirming our theoretical results.

In order to observe the fully-distributed synchronization performance of PISync, we also performed experiments on the 5x4 grid topology. Figure 5 summarizes the maximum synchronization error to the node with identifier 1 for FTSP and fully-distributed PISync, respectively. Even on this topology with a diameter of 9, the exponential increase of the synchronization error of FTSP with the distance can easily be observed. Moreover, although nodes 15 and 20 are neighbors on the grid topology, we observed more or less 10 microsecond maximum error between these nodes. Differently, since sensor nodes synchronize to their direct neighbors PISync, neighboring nodes are more tightly synchronized as compared to far-away nodes. Moreover,

similarly to the flooding-based scenario, the fully-distributed PISync is superior than FTSP in terms of synchronization error.

We also have some observations related to the computation and memory requirements of PISync. First, the implementation of PISync is quite simple that gives rise to the significantly reduced code size for time synchronization. The application code size in our experiments was 18000 bytes for FTSP while it was 15432 for PISync. Besides, since FTSP requires a least-squares regression table to store the time information of the reference node, its RAM requirements is considerably more than that of PISync, which we measured as 52 bytes for FTSP and 16 bytes for PISync, respectively, Finally, at each reference time reception, FTSP performs least-squares regression on the stored information to calculate the estimated regression line. We measured that these operations took approximately 5,5 milliseconds on MICAz platform. On the other hand, since PISync requires only a few arithmetic operations to perform synchronization, it decreases the computation overhead by 50 times as compared to FTSP, which we measured as only 145 microseconds. Our overall impression about the experiments is PISync is superior to FTSP in terms of synchronization error, computation requirements and memory overhead.

VI. Conclusions

We proposed a new control theoretic distributed time synchronization algorithm, named PISync, which is based on a Proportional-Integral (PI) controller. We provided a theoretical analysis in terms of the stability, convergence rate and steady-state error of the proposed algorithm. We highlighted the benefits of this approach as compared to least-squares based time synchronization and presented a practical flooding-based and fully-distributed protocol implementations of it. In the light of the real-world experiments, we observed that PISync has considerably better synchronization performance and scalability over least-squares based time synchronization with the additional advantage of minimal resource requirements.

References

[1] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*. John Wiley & Sons, 2010.

[2] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 39–49.

[3] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler, "Elapsed time on arrival: A simple and versatile primitive for canonical time synchronisation services," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 1, no. 4, pp. 239–251, 2006.

[4] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal Clock Synchronization in Networks," in *7th ACM Conference on Embedded Networked Sensor Systems (SenSys), Berkeley, California, USA*, November 2009.

[5] T. Schmid, Z. Charbiwala, Z. Anagnostopoulou, M. B. Srivastava, and P. Dutta, "A case against routing-integrated time synchronization," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/1869983.1870010

[6] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow flooding in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, p. 1, 2013.

[7] K. Yildirim and A. Kantarci, "External gradient time synchronization in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 633–641, March 2014.

[8] K. S. Yildirim and A. Kantarci, "Drift estimation using pairwise slope with minimum variance in wireless sensor networks," *Ad Hoc Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870512001618

[9] M. Leng and Y.-C. Wu, "Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays," *Signal Processing, IEEE Transactions on*, vol. 59, no. 10, pp. 4860–4870, 2011.

[10] X. Cao, F. Yang, X. Gan, J. Liu, L. Qian, X. Tian, and X. Wang, "Joint estimation of clock skew and offset in pairwise broadcast synchronization mechanism," *Communications, IEEE Transactions on*, vol. 61, no. 6, pp. 2508–2521, 2013.

[11] M. Leng and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *Signal Processing, IEEE Transactions on*, vol. 59, no. 11, pp. 5404–5414, 2011.

[12] J.-M. Berthaud, "Time synchronization over networks using convex closures," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 265–277, Apr. 2000. [Online]. Available: http://dx.doi.org/10.1109/90.842147

[13] J. Chen, Q. Yu, Y. Zhang, H.-H. Chen, and Y. Sun, "Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 6, pp. 2963–2973, 2010.

[14] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Optimal synchronization for networks of noisy double integrators," *Automatic Control, IEEE Transactions on*, vol. 56, no. 5, pp. 1146–1152, 2011.

[15] K. S. Yildirim and Ö. Gürcan, "Efficient time synchronization in a wireless sensor network by adaptive value tracking," *Wireless Communications, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.