Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

High-precision synchronisation in Wireless Sensor Networks with no tuning in the field

Alberto Leva, Federico Terraneo

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo da Vinci, 32 - 20133 Milano, Italy
(e-mail: {alberto.leva,federico.terraneo}@polimi.it)

Abstract:
Many Wireless Sensor Network (WSN) services rely on time synchronisation, and since WSN nodes frequently operate in varying ambient conditions, adequate compensation for thermally-induced clock drift is needed. Also, WSN software – including both applications and operating systems – has to run on heterogeneous hardware, which requires synchronisation solutions that can be deployed with no or minimal tuning in the field. In a previous paper we proposed a joint clock synchronisation and skew/drift compensation scheme entirely realised as a decentralised discrete-time LTI control system, achieving high precision and low-power operation, and compensating thermal drift without temperature measurements. Here, conversely, we concentrate on generality with respect to hardware, and tuning-free deployment. We show that such requirements practically rule out feedforward thermal compensation (which further backs up our previous proposal), devise a methodology to tackle the problem, and present an application demonstrated by experimental tests.

Keywords: Wireless Sensor Networks, time synchronisation, hardware independence, tuning-free deployment.

1. INTRODUCTION

This paper is part of a long-term research, extensively described in Leva et al. [2013], aimed at bringing control-theoretical methodologies and techniques into the *design* – not just the management – of computing systems. A relevant branch of that research concerns the Wireless Sensor Network (WSN) domain, that is nowadays of great importance in many communication, data processing, and control applications. More specifically, we are here concerned with time synchronisation, which is often crucial for critical WSN services.

In a previous work [Leva and Terraneo, 2013], we showed that adopting a control-centric attitude right from the *statement* of the synchronisation problem, brings significant advantages. The quoted work presented a solution named FLOPSYNC, which stands for Feedback LOw Power SYNChronisation, characterised by high precision and low consumption overhead, having a single and easily interpreted design parameter, and counteracting thermal clock drift without the need for temperature measurements. With respect to the previous state of the research, see again Leva and Terraneo [2013] and the papers quoted therein, the additional contribution of this work can be summarised as follows.

- We provide more formal a structure for the idea of separating feedback synthesis and disturbance modelling. We consequently propose a means to quantify the relative importance of the various disturbance sources, based on *nominal* data on the target hardware and the expected operating conditions.
- As a direct consequence of the above analysis, we show that attempting to counteract thermal drift by feedforward compensation based on temperature measurements, in general does not provide reliable and robust enough results, hence not paying back for the incurred cost in terms of control algorithm complexity and tuning difficulties.
- We devise a methodology to synthesise a synchronisation controller designed along the approach of Leva and Terraneo [2013], employing again only nominal data. This allows for an effective and robust tailoring of the overall system to any target hardware and expected operating conditions, without the need for tuning in the field.
- We provide an application example for the methodology just mentioned, based on a controller structure that extends those in the quoted reference.

Experimental tests are reported and synthetically commented on, to back up the overall proposal.

2. RELATED WORK AND CONTRIBUTION

As stated in the definition of the Network Time Protocol, "the *offset* of two clocks is the time difference between them, while the *skew* is the frequency difference (first derivative of offset with time) between them. Real clocks exhibit some variation in skew (second derivative of offset with time), which is called *drift*". In addition to the above, here we also account for short-term frequency variations, which NTP does not mention, and are commonly called *jitter*.

The mainstream literature approach divides the time synchronisation problem into two subproblems: *clock synchronisation*, which is making the nodes' clocks periodically agree with a reference one, and *skew compensation*, which is maintaining said agreement from a clock synchronisation, with no information

Copyright © 2014 IFAC

on the reference time past that event, till the subsequent one. Both problems are typically addressed via a periodic transmission of *timestamps* from the reference node.

Several works, especially at the dawn of WSN research, were concerned essentially with clock synchronisation. For example, in DMTS [Ping, 2003] local clocks are periodically overwritten with a broadcast time reference, while in TPSN [Ganeriwal et al., 2003] nodes synchronise pairwise based on timestamp exchanges. More or less at the same time, however, the need for skew compensation emerged, leading to schemes like RBS [Elson et al., 2002], that uses skew estimates obtained by linear regression on past errors, and FTSP [Maroti et al., 2004], that concentrates on the efficient distribution of synchronisation packets. More recently, the research focus was extended to improving the quality of the individual hardware and software components of synchronisation schemes. Notable examples are glossy [Ferrari et al., 2011], that proposes an efficient *flooding scheme* to have synchronisation packets reach all the nodes with a transmission delay determinism in the sub-microsecond range, and TCTS [Schmid et al., 2009], that enhances FTSP with thermal compensation using a temperature-to-frequency crystal model.

The different (control-centric) perspective of FLOPSYNC introduced three main novelties. First, timestamp transmission is not required unless at boot time, as the synchronisation-related information is drawn from packet arrival times. Second, clock synchronisation and skew compensation are treated jointly. Third, thermal upset is counteracted by feedback only. The result is a decentralised scheme composed of one discrete-time LTI controller per node, having a single parameter related to the desired error convergence rate.

Despite the so achieved relevant simplification and performance enhancements, some questions are however open. The opportunity of including feedforward compensation for thermal upset needs investigating. Although quite reasonable a default value for the control parameter was proposed, a procedure is worth devising to select it in a more hardware-tailored manner without field experiments. In the case of highly variable ambient condition, a scheduling mechanism for that parameter is to be studied. Finally, since the control-based approach of Leva and Terraneo [2013] could obviously give rise to different control structures for specific purposes, the *modus operandi* just envisaged for configuration and deployment should be defined as generally as possible and convenient.

3. BACKGROUND

According to [Leva and Terraneo, 2013, Section III] and slightly adapting the notation for the particular purpose of this work, the control scheme for tackling the joint synchronisation/compensation problem at the generic i-th node via FLOPSYNC, is that of Figure 1.

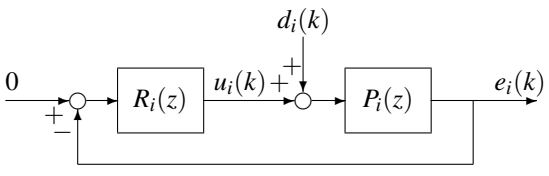

Fig. 1. Control scheme for the i-th node.

Denoting with t the reference time, with $t_i(t)$ the local time of the i-th node, and with k an integer time index counting the clock synchronisation events, that occur with a period T known to all the nodes, FLOPSYNC measures the synchronisation error $e_i(k) := t_i(t(k)) - t(k)$ at the *end* of each synchronisation period – thus joining clock synchronisation and skew compensation naturally – as the difference, counted in the local clock, between the *expected* and the *actual* arrival time of the k-th synchronisation packet. Then, to compensate for that error, FLOPSYNC adjusts the time to wait before the $(k+1)$-th packet (i.e., implicitly, its expected arrival time) to be $T+u_i(k)$, where $u_i(k)$ is the control signal. This simply means that the next expected arrival time is set to the current one, plus T, plus u_i. All the possible error causes are thus collectively represented by the disturbance $d_i(k)$, and the model of the controlled system is just

$$E_i(z) = P_i(z)\big(U_i(z) + D_i(z)\big), \quad P_i(z) = \frac{1}{z-1}. \tag{1}$$

The FLOPSYNC controller can take several forms, as discussed in Leva and Terraneo [2013], but in any case is extremely simple given the form of $P_i(z)$ in (1). Here we consider a possibility not included in the quoted reference, i.e.,

$$R_i(z) = \frac{3(1-\alpha)z^2 + 3(\alpha^2-1)z + 1 - \alpha^3}{(z-1)^2}, \tag{2}$$

that produces

$$\frac{E_i(z)}{D_i(z)} = \frac{(z-1)^2}{(z-\alpha)^3}, \tag{3}$$

thus asymptotically rejecting ramp-like disturbances. The results shown later on apply to any integral-endowed controller with a single parameter related to the desired error convergence rate, however. The key point is that such controllers guarantee stability and zero steady-state error – in the specific case of (2), the constraint $0 < \alpha < 1$ is introduced to also avoid a closed-loop oscillatory mode – in a hardware-independent manner, while α governs the error convergence in relative terms, i.e., the number of steps required to reduce it to a given fraction of its peak value following a disturbance. If however one wants to impose *absolute* error bounds, the addressed hardware and operating conditions come into play.

It is also worth stressing that the control is computed with period T, and for the rest of the time the system operates in open loop. If an application or the operating system require an estimate of $\hat{t}$ of the reference time amidst two clock synchronisations, this is computed as

$$\hat{t} = kT + (t_i - t_i(k))\frac{T}{T + u_i(k)} \tag{4}$$

where k is the index of the last synchronisation (possible initial time offsets are eliminated at boot time with a procedure inessential to discuss herein). Thus, continuous-time physical (e.g., thermal) phenomena with a time scale comparable or faster with respect to T yield an error that cannot be counteracted until the next synchronisation, and may be of concern.

Summing up, the approach followed in this research has the peculiarity and the advantage that the behaviour of the loop is independent of any hardware and operating condition information, since these are all modelled as the source of $d_i(k)$. As a result, then,

- qualitative control requirements (e.g., the ability to asymptotically reject a certain disturbance) can be assessed by choosing a convenient structure for $E_i(z)/D_i(z)$,
- relative error convergence bounds just require to choose α in a hardware- and operation- independent manner,
- while absolute error bounds (e.g., on its peak value and/or the time to fall below a certain magnitude) can be enforced based just on nominal hardware and operating condition data, without jeopardising stability as the required analysis only concerns the disturbance generation mechanism.

4. DISTURBANCE ANALYSIS AND MODELLING

Denoting with f_o the clock frequency of the reference node, assumed constant as it provides the reference (universal) time t, and with $f_i(t)$ that of the i-th node, $d_i(k)$ takes the form

$$d_i(k) = \int_{t_i(k)}^{t_i(k+1)} \frac{f_i(\tau) - f_o}{f_o} d\tau. \tag{5}$$

Note that we expressed the integration interval limits in the node time t_i for consistence with the FLOPSYNC algorithm operation, but this is substantially equivalent to using the universal time t, because the synchronisation error has to be in the order of milliseconds or less, while the synchronisation period ranges from a few tenths of seconds up to minutes.

Writing now $f_i(t) = f_o + \delta f_{oi} + \delta f_{si}(t) + \delta f_{ji}(t)$, where δf_{oi} accounts for differences in the crystal nominal frequencies, $\delta f_{si}(t)$ for frequency variations that produce the node clock time-varying skew – thus comprising drift – and δf_{ji} for the shorter-term frequency fluctuations that result in jitter, we have

$$d_i(k) = \frac{\delta f_{oi}}{f_o}\big(t_i(k+1) - t_i(k)\big) + \int_{t_i(k)}^{t_i(k+1)} \frac{\delta f_{si}(\tau) + \delta f_{ji}(\tau)}{f_o} d\tau. \tag{6}$$

We assume the same f_o for all nodes to simplify the notation, as relaxing this assumption is straightforward. The first term in (6) can be considered a constant of value $\delta f_{oi} T/f_o$, as in any sane realisation $t_i(k+1) - t_i(k)$ will be kept very close to the synchronisation period. Said term is thus eliminated by the integral action of $R_i(z)$, and for the purpose of transient error behaviour we can concentrate on the skew and jitter ones, i.e.,

$$d_i^s(k) = \int_{t_i(k)}^{t_i(k+1)} \frac{\delta f_{si}(\tau)}{f_o} d\tau, \quad d_i^j(k) = \int_{t_i(k)}^{t_i(k+1)} \frac{\delta f_{ji}(\tau)}{f_o} d\tau. \tag{7}$$

The jitter term comes from oscillator nonlinearities and electrical phenomena. It is in general of modest entity, and in any case clearly provides the endpoint for the precision of time estimates between clock synchronisations. As for the loop, this term appears as a small and fast disturbance, thus only requiring to limit the high-frequency control sensitivity magnitude to avoid useless system upset—a first clue for selecting α.

The skew (and drift) term conversely depends basically on thermal phenomena, therefore being the most important to consider herein. To this end, we assume for the crystal temperature-to-frequency relationship the standard form Nakazawa et al. [1979]

$$f_i(t) = f_o \left(1 - \frac{\beta_i}{10^6}(\theta_i(t) - \theta_o)^2\right) \tag{8}$$

where θ_i is the crystal temperature, θ_o that corresponding to the nominal frequency (we also assume the same θ_o for all nodes, see above for the reason), and $\beta_i > 0$ a crystal-specific parameter expressed in $ppm/^\circ C^2$. It is worth noticing that some works introduce a small linear term in (8), and also suggest to account for some thermal hysteresis, see e.g. Marchetto et al. [2012]. However, both the phenomena just mentioned are hardly ever mentioned in crystal datasheets, and no parameters on them are normally provided. As, such, we decided to stick to (8) for applicability reasons.

Coming back to the main subject, we have thus

$$\delta f_{si}(t) = -\frac{\beta_i}{10^6}(\theta_i(t) - \theta_o)^2 f_o. \tag{9}$$

and consequently

$$d_i^s(k) = -\frac{\beta_i}{10^6} \int_{t_i(k)}^{t_i(k+1)} (\theta_i(t) - \theta_o)^2 dt. \tag{10}$$

The main problem with $d_i^s(k)$ is the wide variety of thermal dynamics encountered in WSN nodes, depending on the size and shape of the circuitry, the location of the temperature sensor, the characteristics of the casing, the type of installation, and the ambient conditions. The following section discusses the matter in a view to evidencing the potentially excessive criticality of feedforward temperature compensation.

5. CRITICALITY OF TEMPERATURE COMPENSATION

WSN nodes are typically compact devices, where power considerations often oblige to reduce hardware to the bare essential. As such, if the devised controls aim at a widespread deployment, it is reasonable to assume that the only temperature sensor available is that on the node processor. Said processor is mounted on a circuit board, normally near to the clock quartz, and inside a casing. Neglecting thermal diffusion in the circuitry and assuming that currents are so low for Joule self-heating to be negligible as well, heat mainly flows from the external ambient to the node casing, then to the internal air, and then to both the processor and the crystal, more or less in parallel. This allows for a first-cut description of the thermal phenomena of interest as the continuous-time LTI model

$$\begin{cases} Cc\dfrac{d\theta_c(t)}{dt} = G_{ec}\big(\theta_e(t) - \theta_c(t)\big) - G_{ca}\big(\theta_c(t) - \theta_a(t)\big) \\ Ca\dfrac{d\theta_a(t)}{dt} = G_{ca}\big(\theta_c(t) - \theta_a(t)\big) - G_{ap}\big(\theta_a(t) - \theta_p(t)\big) \\ \qquad\qquad -G_{ax}\big(\theta_a(t) - \theta_x(t)\big) \\ Cp\dfrac{d\theta_p(t)}{dt} = G_{ap}\big(\theta_a(t) - \theta_p(t)\big) \\ Cx\dfrac{d\theta_x(t)}{dt} = G_{ax}\big(\theta_a(t) - \theta_x(t)\big) \end{cases} \tag{11}$$

where $C_{c,a,p,x}$ and $\theta_{c,a,p,x}$ are the thermal capacities and the temperatures of casing, internal air, processor and crystal, $\theta_e(t)$ is the (exogenous) external temperature, and G_{jk} the thermal conductance between the two elements denoted by the subscripts $j,k \in \{e,c,a,p,x\}$, with the convention just introduced.

There is not the space here to report an extensive simulation campaign with model (11), hence we just sketch out the methodology we followed, and the outcome we obtained. To take a transient as the reference for evaluating thermal stress rejection, we assumed that the external temperature varies in a time span of T by the maximum amount $\Delta\theta_e$, starting from a condition with the system in thermal equilibrium. We then fed the mentioned input to (11) with different initial values of the external temperature, analysing the consequent behaviour

of the measured temperature θ_p and the disturbance one θ_x to be compensated for. It was straightforward to observe (and the reader can easily verify) that on the time horizon of a few synchronisation periods, the effect of just slightly mistaking some conductance or capacity (where "slightly" means by an amount compatible with the variability of a node produced in series, or with the effects of unforeseen conditions like humidity) can be so relevant to make feedforward compensation of hardly any use, if not detrimental. Of course the problem is less relevant on longer time scales, but given the requirements typically imposed on feedback synchronisation control – see again Leva and Terraneo [2013] – either feedforward acts within a few periods at most, or it is of no practical use.

To further support the idea above, consider as a representative example the temperature compensation method proposed in Schmid et al. [2009]. The main idea behind that work is to collect temperature data along the WSN operation, that each node uses to progressively construct its own temperature-to-frequency oscillator model. Quite intuitively, such a mechanism is suitable for synchronisation/compensation schemes base e.g. on regression, thus aiming at a control time scale that is inherently slower than that quantified by the regression window, which spans several periods. However, the same compensation method could not act at the time scale that is required to be effective with the tighter control time scale – thus the strong feedback – of FLOPSYNC.

Summarising, when feedback schemes adopting a FLOPSYNC-like approach are used and thermal stress on a wide operating range is of concern, the only reasonable approach from an engineering standpoint is to draw clues for selecting both α and T by constraining the peak value and the recovery time of the synchronisation error after a thermal event, based on information on the expected operating temperature range, and the maximum variation rate of the same temperature. This can be used to configure the node offline from nominal data, as illustrated in the following.

6. PARAMETER SELECTION

This section deals with the selection of the synchronisation period T and of the convergence-related single controller parameter (α in the FLOPSYNC case) based on nominal information on the crystal oscillator, operating condition data and error bounds, namely (we drop the i subscript for simplicity as the analysis applies to all the nodes in a WSN)

- the nominal crystal temperature to frequency parameters, θ_0 and β, which can be obtained from the crystal data sheet,
- a minimum external temperature $\theta_{e,min}$ and a maximum one $\theta_{e,max}$, such as the minimum temperature during winter and the maximum during summer for outdoor applications,
- a maximum magnitude $r_{\theta,max}$ for the temperature variation rate,
- a maximum magnitude $\Delta\theta_{max}$ for the temperature swing in a single thermal event, as it is very unlikely that said swing spans the entire $(\theta_{e,min}, \theta_{e,max})$ range,
- a maximum magnitude e_{max} for the tolerable peak value of the controlled error (i.e., that measured at synchronisation instants) after a temperature event,

Table 1. Design specifications for the two presented cases

	case 1	case 2	
β	0.025	0.04	$ppm/^\circ C^2$
θ_0	25	25	$^\circ C$
$\theta_{e,min}$	-20	15	$^\circ C$
$\theta_{e,max}$	50	22	$^\circ C$
$\Delta\theta_{max}$	25	5	$^\circ C$
$r_{\theta_{max}}$	8	0.5	$^\circ C/min$

- a maximum time $t_{r,max}$ (a multiple of T given the assumed setting) for the magnitude of the same error to fall below a prescribed value $\bar{e} < e_{max}$ after such an event.

Of course the purpose is to devise conservative bounds, conceived however in such a way that they can be obtained in a straightforward manner, to the advantage of quick system setup and interpretability on the part of the user.

To start the analysis, suppose that the node is initially at a thermal equilibrium with the external temperature $\theta_{e,min}$, and then that its temperature θ increases up to $\theta_{e,min} + \Delta\theta$ with an exponential transient, starting from $t = 0$ for simplicity, and characterised by a maximum rate $r_{\theta,max}$, i.e.,

$$\theta(t) = \theta_{e,min} + \Delta\theta_{max}\left(1 - e^{-t\frac{r_{\theta,max}}{\Delta\theta_{max}}}\right) \tag{12}$$

This *stimulus*, together with the one from $\theta_{e,max} - \Delta\theta_{max}$ to $\theta_{e,max}$, and the ones with reversed start and end temperatures (the need for all four comes from the generally asymmetric position of $\theta_{e,max}$ and $\theta_{e,min}$ with respect to θ_o) are the harshest possible ones compatibly with the assumed conditions, although being realistic enough not to yield too conservative results, as they could be caused by a radiative power step (think of a mobile node entering or exiting a tunnel, for example). Supposing – to make the analysis even more worst-case – that the temperature increase starts immediately after a synchronisation instant, corresponding to $k = 0$ for the same reason above, and that the crystal temperature instantaneously follows the external one, we have

$$d^s(k) = -\frac{\beta}{10^6}\int_{kT}^{(k+1)T}(\theta(t) - \theta_o)^2\,dt. \tag{13}$$

Once the crystal (i.e., β and θ_o) and the operating condition limits (i.e., $\theta_{e,max}$, $\theta_{e,min}$, $\Delta\theta$ and $r_{\theta,max}$) are known, four application of (13) for the aforementioned worst case temperature changes permits to calculate the worst-case disturbances $d^s_{1,2,3,4}(k)$ for a single value of T. Then, feeding $d^s_{1,2,3,4}(k)$ to the dynamic system (3), initialised at the convenient equilibrium, readily provides the maximum error magnitude and the number of steps needed to recover $\bar{e}$ for a given value of α.

Summarising, with a very reasonable (offline) computational effort, one can obtain two surfaces that given β, θ_o, $\theta_{e,max}$, $\theta_{e,min}$, $\Delta\theta$ and $r_{\theta,max}$, provide the peak error and the recovery time in steps. Two examples are reported in Figures 2 and 3, named "case 1" which assumes quite a wide and severe operating range, and "case 2" which refers to a smoother situation like e.g. that of an indoor node, but a crystal with a higher frequency dependence on temperature (recall that datasheets provide a worst-case value).

Suppose now that the specifications require a certain e_{max} and $t_{r,max}$ to recover a given $\bar{e}$, that we invariantly set to $20\,\mu s$. The results of Figures 2 and 3 can easily be used to determine the

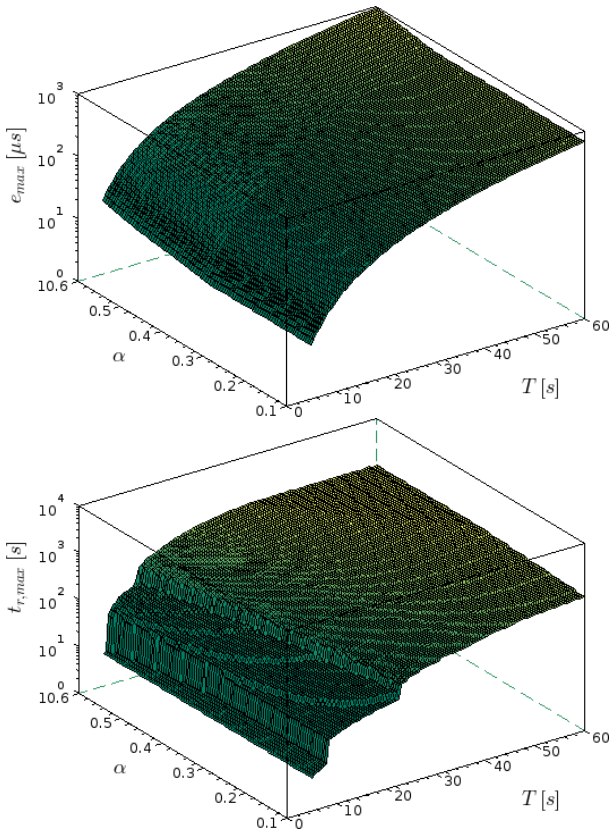

Fig. 2. Peak error (top) and recovery time (bottom) as functions of T and α – case 1.

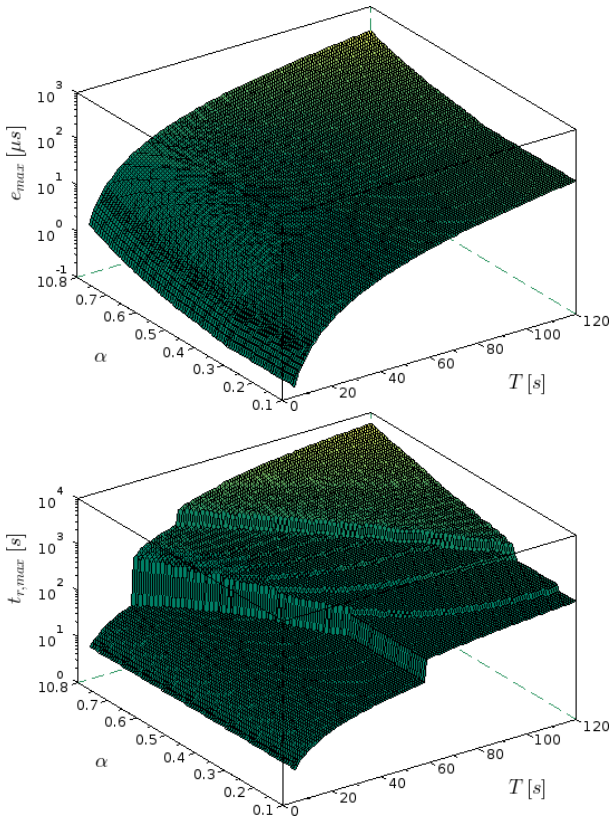

Fig. 3. Peak error (top) and recovery time (bottom) as functions of T and α – case 2.

feasible (T, α) couples, as indicated by the green regions in Figures 4 and 5 (the observed quantisation effects are due to the synchronisation period granularity). Consistently with the nature of cases 1 and 2, in the latter more stringent requirements were formulated, to show the effectiveness and flexibility of the proposed methodology.

Given the results exemplified in Figures 4 and 5, as a further aid to finalise the selection of a (T, α) couple, one can consider the high-frequency magnitude of the frequency response of (3), i.e., with the notation of this section,

$$\left|\frac{E(e^{j\vartheta})}{D(e^{j\vartheta})}\right|_{\vartheta=\pi} = \frac{4}{(1+\alpha)^3} \tag{14}$$

and observe that lower values of α allow for larger synchronisation period at the cost of some amplification of high-frequency jitter. Therefore, the red curve at the edge of the feasible region of Figures 4 and 5 can be viewed as sort of a Pareto curve for the quality of synchronisation (high α, low jitter) versus power consumption (large synchronisation period T) tradeoff.

We also created a configuration tool to easily enforce the required constraints: by filling in the form of Figure 6, the user is presented with an output similar to Figures 4 and 5, from which T and α are easily selected. The items marked A, B and C in Figure 6 reflect in the obtained behaviour as shown by Figure 7. The tool, written in Scilab, will soon be released as free software.

To end the section, a final remark is in order. The results presented so far are very natural to use with FLOPSYNC. However their interest is more general, thus making it legitimate to wonder how wide their applicability to other synchronisation schemes is. From this viewpoint, it is worth dividing the existing methods in two broad categories. The first one, comprising FLOPSYNC, contains all methods – like for example

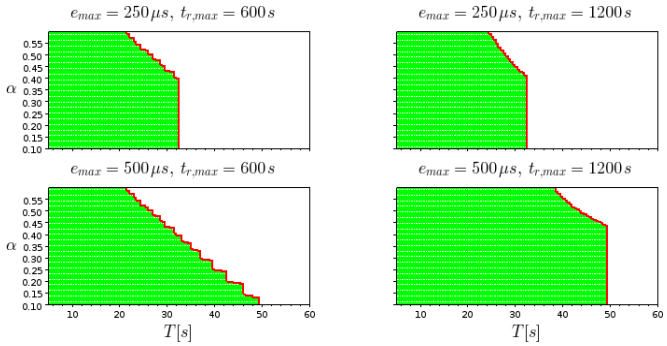

Fig. 4. Feasible (T, α) couples – case 1.

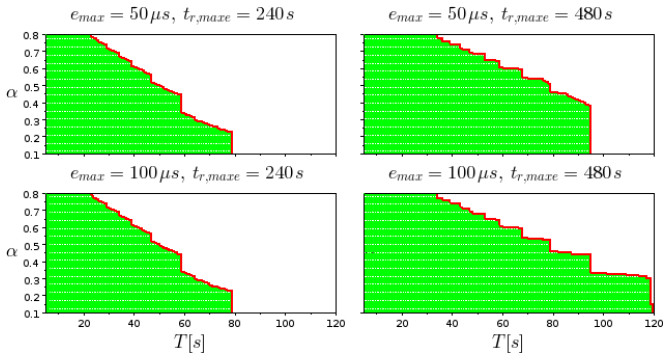

Fig. 5. Feasible (T, α) couples – case 2.

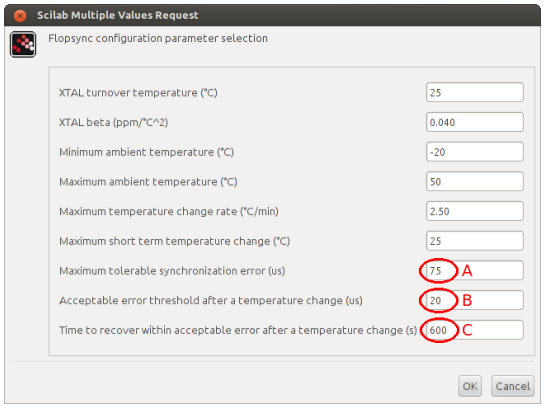

Fig. 6. The FLOPSYNC configuration tool interface.

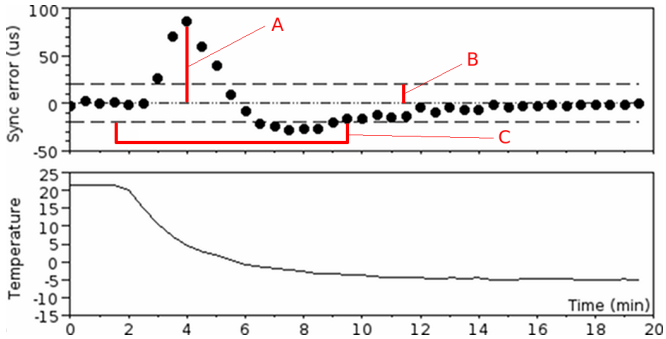

Fig. 7. Synchronisation error during a temperature change (see Figure 6 for the meaning of A, B and C).

FBS [Chen et al., 2010] – that make explicit use of a feedback controller for clock synchronisation. In such cases, one can take the proposed synthesis method, compute the peak error magnitude and the time required to reduce it below the $\overline{e}$ of choice as a function of the controller parameter(s), and use the so obtained result as a means for selecting said parameter(s) in a manner analogous to that proposed herein. The second category contains all other methods, that either do not have any control parameters (thus not being suitable for generalising the proposed ideas) or encompass some configuration that has quite indirect a relationship with error convergence, like for example the length of the regression window in FTSP [Maroti et al., 2004]. These form in some sense the boundary of the set of methods to which the proposed approach is applicable, and for them the achievable benefits need further investigation.

7. EXPERIMENTAL RESULTS

To test the presented parameter tuning method, the FLOPSYNC controller was implemented in a real WSN node platform built from COTS components, namely STM32vldiscovery development boards and nRF24L01-based 2.4GHz radio transceivers and exposed to temperature changes to measure the synchronisation error.

Due to space limitations only a single test is here reported, where the controller parameter $\alpha = 3/8$ and $T = 30s$ were selected, according to the specifications of "case 1" with a tolerable maximum error of $250\mu s$ and a desired time to recover within $20\mu s$ of 10 minutes. The node was then exposed to a temperature change of more than $25\,^{\circ}C$ with a rate of $8\,^{\circ}C/min$. The maximum observed synchronisation error is $86\mu s$, and as expected is lower than the design parameter of $250\mu s$. For what concerns the settling time to within $20\mu s$, it is 7 minutes, again within the design constraints.

8. CONCLUSIONS AND FUTURE WORK

A study was presented on how to make synchronisation schemes for wireless sensor networks free from tuning in the field, in a view to ease their deployment and reduce performance variability. If the considered scheme is designed with a fully control-theoretical approach, as is the case with FLOPSYNC, it is quite straightforward to determine its parameters offline, based on nominal hardware data and the expected operating conditions. If on the contrary the considered scheme is less keen to be formalised as a feedback control one, things are a bit more complex, but from a conceptual viewpoint the presented results should be (at least in part) applicable anyway, although further research on the matter is in order.

As a result, future work will address an extensive verification of the applicability limits here sketched for the proposed approach, more extensive experimental campaigns and their analysis, particularly considering harsh ambient conditions, considerations on the possible opportunity of devising some parameter scheduling method, and the integration of the so extended FLOPSYNC scheme (and its tuning *modus oeprandi*) in standard WSN communication protocols.

REFERENCES

J. Chen, Q. Yu, Y. Zhang, H. Chen, and Y. Sun. Feedback-based clock synchronization in wireless sensor networks: A control theoretic approach. *IEEE Transactions on Vehicular Technology*, 59(6), 2010.

J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Symposium on Operation Systems Design and Implementation*, 2002.

F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Information Processing in Sensor Networks (IPSN)*, 2011.

S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *International Conference on Embedded Networked Sensor Systems*, 2003.

A. Leva and F. Terraneo. Low power synchronisation in wireless sensor networks via simple feedback controllers: the FLOPSYNC scheme. In *Proc. 2013 American Control Conference*, pages 5017–5022, Washington, DC, USA, 2013.

A. Leva, M. Maggio, A.V. Papadopoulos, and F. Terraneo. *Control-based Operating System Design*. IET, London, UK, 2013.

P. Marchetto, A. Strickhart, R. Mack, and H. Cheyne. Temperature compensation of a quartz tuning-fork clock crystal via post-processing. In *IEEE International Frequency Control Symposium (FCS)*, pages 1–4, 2012.

M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Conference On Embedded Networked Sensor Systems*, 2004.

M. Nakazawa, Y. Nakamura, and S. Miyashita. Frequency-temperature characteristics of quartz crystal flexure bars and quartz crystal tuning forks. *IEEE Transactions on Sonics and Ultrasonics*, 26(5):369–376, 1979.

S. Ping. Delay measurement time synchronization for wireless sensor networks. In *Intel Research*, 2003.

T. Schmid, Z. Charbiwala, R. Shea, and M. Srivastava. Temperature compensated time synchronization. *IEEE Embedded Systems Letters*, 2009.