Available online at www.sciencedirect.com
ScienceDirect
Ad Hoc Networks 00 (2022) 1–18
Ad Hoc Networks

A neural approach to synchronization in wireless networks with heterogeneous sources of noise

Maurizio Mongelli[1,*], Stefano Scanzio[2]

Institute of Electronics, Computer and Telecommunication Engineering, National Research Council of Italy

Abstract

The paper addresses state estimation for clock synchronization in the presence of factors affecting the quality of synchronization. Examples are temperature variations and delay asymmetry. These working conditions make synchronization a challenging problem in many wireless environments, such as Wireless Sensor Networks or WiFi. Dynamic state estimation is investigated as it is essential to overcome non-stationary noises. The two-way timing message exchange synchronization protocol has been taken as a reference. No a-priori assumptions are made on the stochastic environments and no temperature measurement is executed. The algorithms are unequivocally specified offline, without the need of tuning some parameters in dependence of the working conditions. The presented approach reveals to be robust to a large set of temperature variations, different delay distributions and levels of asymmetry in the transmission path.

Keywords: Clock synchronization protocols, Dynamic state estimation, Two-way timing message exchange, WSN, WiFi

1. Introduction

Clock Synchronization Protocols (CSPs) have a fundamental role in many technological contexts in which a common time reference is required [1]. For example, synchronization is used in Wireless Sensor Networks (WSNs) [2], localization [3, 4], home automation [5], industrial networks [6], traffic scheduling [7, 8], and in a number of other contexts in which actuation and/or sensing must be synchronous. The quantities measuring the asynchronism between the clocks of two nodes in a network are: the *offset*, i.e., the difference between the two clocks and the *skew*, i.e., the normalized difference between the Crystal Oscillator (XO) oscillation frequency and its nominal frequency. The variable component of the skew is the *drift*. Their precise estimation defines the target of the CSP and they are jointly optimized [9]. They typically represent the state of the synchronization problem, when it is formulated under dynamic state equations.

The estimation process may be severely compromised by a number of factors. The most important are: the random delays affecting the communication path between nodes, including software or hardware delays inside them, the precision of nodes in timestamping events, and changes in the environment conditions. Estimation of offset and skew may be driven by signal processing techniques, which assume a time-fixed state (see, e.g., [10] for WSNs) or by dynamic state tracking, e.g., through Kalman filtering (as an example, see [11] for IEEE 1588 protocol). Addressing time-varying conditions means to follow instantaneous fluctuations due to non-stationary noises, such as temperature variations of XOs [12, 13, 14].

1.1. Background and objectives

In the present paper, we study how to compensate with a single technique all the possible factors affecting the synchronization quality. The idea to analyze and compensate a number of causes together is not new. Algorithms derived from machine learning (e.g., neural networks, support vector machines,...) are typically exploited to model complex processes, in the case a theoretical model is not known or it cannot be parameterized in practice because too many measurements of the real system are needed for a satisfactory characterization. The latter is the case of synchronization protocols. For example, the oscillation frequency of an XO is influenced by several environmental factor: the temperature, the supply voltage, vibrations, age, etc. All

*Corresponding author.
Email addresses: maurizio.mongelli@ieiit.cnr.it (Maurizio Mongelli), stefano.scanzio@ieiit.cnr.it (Stefano Scanzio)
[1]Area della Ricerca di Genova, via De Marini 6, Genova, Italy.
[2]Corso Duca degli Abruzzi 24, 10129 Torino, Italy.

these factors, well documented in the scientific literature [12], have not the same influence on the behavior of different types of XOs, and even the same type of XOs differently reacts to environmental conditions, depending on its manufacturing process. As a consequence, to compensate all these factors, each XO must be experimentally characterized with respect to the physical phenomena that can modify its behavior. Such a kind of analysis can only be performed during the manufacturing process of the component, because XOs are usually soldered on the motherboard. All these physical quantities must then be sensed at runtime for their relevant compensation. This last step is not easy and often it is not feasible because, for example, XOs do not usually include temperature sensors. In this case, the estimation must be derived by using sensors in the proximity of the XO. On the other hand, XOs that automatically compensate some external factors exist, but they are hardly integrated in commercial devices since they cost too much. The same difficulties in finding a correct model apply also to other quantities such as the timestamps precision and accuracy, and asymmetric delays. They depend on the hardware, but in the case of software timestamps also on the interference caused by other processes executed in the operating system of the node. For these reasons, we decide to focus on an algorithm that compensates all these aspects together. Some results concentrate the attention on temperature, because it is the most affecting environmental factor.

The algorithm we pursue should be capable to work with minimal online adjustment of the parameters, thus avoiding the need of reconfigurations following the actual behavior of the noises.

Synchronization is formulated as a dynamic state tracking problem beyond regular LQG hypoteses[3] because temperature measurement noise may not always be Gaussian in practical systems [13, 14, 15]. The inherent optimal estimation filter may be hardly derived in closed form. This approach typically has consequences in terms of numerical analysis with complex operations (see, e.g., the Particle Filtering in [10]), which are not easily applicable in devices in which computational power or energy are scarce resources. Since the investigated suboptimal filter is based upon neural approximation, the approach may lead to a heavy computational effort in the offline phase (during which the training of the neural network is provided), but synchronization corrections are provided online almost instantly. Delay assymetry is also addressed jointly with temperature variations. This avoids configuring countermeasures to assymetry that are separate from the rest of the synchronization scheme.

1.2. Contribution

The method firstly outlined in [15] for *receiver-receiver* CSPs [6], is now applied in the sender-receiver context, more used in practice, and under realistic conditions of WSN and WiFi networks, including delay asymmetry. Despite the considered CSP drives delay compensation, we show how no knowledge of delay is necessary for the used estimation techniques. An enhancement of the method is proposed to cope with exponentially distributed delays, a condition not often detectable in practice, but analyzed in some scientific works [16]. The method provides good generalization capabilities to different delays distributions (i.e., Gaussian and exponential delays). The multi-hop context is also addressed to limit computational cost and simplify the applicability of the method.

1.3. Organization of the paper

The paper is organized as follows. The next section deals with the analysis of the state of the art and highlights the position of the present paper. Section 3 addresses the mathematical formulation of the estimation problem. The subsequent sections enter in the details of the estimation techniques proposed, including computational and implementation aspects. Section 9 defines the setting of the experiments and Section 10 discusses the results. Conclusions and future work are finally outlined at the end of the paper.

2. Related literature

2.1. State estimation

Dynamic state estimation for synchronization is an open issue for environments with non-Gaussian and non-stationary noises [13, 14]. An example for WSN has been reported in [10], by introducing Particle Filtering (PF). [10] shows how addressing time-varying conditions may considerably improve the synchronization gain over signal processing techniques. PF is able to adapt to Gamma distributed delays better than signal processing, which works well under Gaussian or exponential delays. PF belongs to the optimal Bayesian framework for dynamic state estimation. This is exactly the research line we want to pursue here, without incurring in the computational burden involved by PF.

As far as signalling processing techniques are concerned, our approach has been compared with [17], which is a reference target in this field (see, e.g., [16]), since it presents a computationally light approach, which is also robust to the underlying network delay density function and asymmetry. More refined techniques are available as well, for example, in the presence of exponentially distributed delays [16].

2.2. Parameters setting

Online adaptation may be critical if the statistical parameters of the noises cannot be known in advance. More specifically, the covariance matrix of the noises is typically used as a parameter of the mentioned algorithms (Kalman [11], signal processing as in [16] and Particle Filtering (PF) in [10]). How parameters setting may be a critical task in Kalman is evidenced by [18], in which practical guidelines are provided. This critical aspect has been also registered by [14], in which the parameters of the estimation algorithm are tuned online and by [13], in which the parameters of the temperature-skew mapping are supposed to be known in advance. Synchronization solutions

[3]Linear dynamics of the system, quadratic cost function and Gaussian noises.

with self-learning capabilities may be hardly found in the literature. [19] has recently investigated how to adapt the time window of linear regression. The approach has been tested in stationary Gaussian conditions.

2.3. *Temperature noise*

Recent works address synchronization in WSNs by overcoming the temperature noise. In [13], the thermal drift is removed in advance, by exploiting the relationship between XO frequency and the temperature. A multi-model Kalman filter is studied in [14] to obtain the model likelihood for the skew, based on the measured temperature. The main advantage of the two approaches relies on the possibility to reduce the sending rate of synchronization messages, by keeping unchanged the synchronization quality since the temperature is locally compensated. An ARMAX model is studied in [20] to compensate temperature and aging effects. An upper bound of the error is derived in closed-form under Gaussian assumptions. The mentioned works rely on a mapping table from temperatures to clock skews [21]. [21] models the correlation between clock skews and temperature variations through the least squares method, thus achieving more flexibility, still relying on temperature measurements. The approach presented here does not exploit any measurement of the temperature. [22] deals with high latency networks by introducing a new message exchange in two steps: in the first one the delay is estimated and, in the second one, Kalman is applied. The refined procedure reveals to be robust to noise, including temperature changes.

2.4. *Asymmetric Delays*

Despite [17, 16] do not address delay asymmetry explicitly, they reveal to be robust to several working conditions, including asymmetry. More recent works address the mitigation of delay asymmetry [23, 24, 25, 26, 27] explicitly. Timestamping corrections are provided to compensate the synchronization error induced by asymmetry. [23] requires additional messages in the protocol. [24, 25] exploit different kinds of link speed measurements to infer the level of asymmetry. As evidenced in [26], those measurements may be not always sufficient if the internal delays of the device have a predominant role. Proper statistical information is derived from additional link/internal device delays [26]. In the very rare case of intermediate devices without the compensation of the packet residence time inside the device, traffic queues can be measured [27]. The inherent corrections in [26] may require an accurate setup of the devices. [28] and [29] apply the Boot-strap method under the assumption of Gamma and exponential distributed bias in asymmetry, respectively. A similar approach is applied to Pareto distributed delays in [30]. The robustness of the methods are accurately analyzed with respect to parameters of the probability distributions. An important advantage of [28] consists of the simple calculations executed to derive the bias estimation. In [27], the corrections may be sensitive to parameters changes (e.g., size of the observation window) and an accurate analysis is needed for them. Here, the correction is derived without any additional measurements or knowledge of the device and it is applied jointly to the rest of the compensation steps.

Table 1. Topics of research in synchronization and state of the art.

	temp.	asym.	unknown noises	real applic.
[9]				●
[11]				○
[13]	○			●
[14]	○			
[21]	○		●	○
[17]		○	●	●
[19]			●	
[15]	●		●	○
[16]		○		●
[18]			○	○
[22]	●			●
[20, 23–29]		●	○	○
Present paper	●	●	●	○

Although in practice the communication channel is sufficiently symmetric for the majority of the applications to not affect too much the synchronization quality, this consideration does not hold for in-node latencies, i.e., the delay inside the nodes between the sending/reception of a packet and the acquisition of the relevant timestamp exploited in clock correction. This problem also applies to such nodes that acquire the timestamp in hardware, and its effect on synchronization quality is clearly amplified if the network contains heterogeneous nodes. From the viewpoint of the synchronization protocols, in-node or communication channel asymmetries are indistinguishable and they have exactly the same consequence on the achievable synchronization quality. This evidence will be analyzed in detail in subsection 9.1, which is based on data derived from scientific literature and acquired from real devices.

2.5. *Position of the paper*

Table 1 summarizes the discussion presented in this section and highlights the contribution of the present paper. A • mark is assigned if the paper exactly addresses the topic of interest. A ◦ mark is assigned if the topic is partially addressed; for example, a ◦ mark is assigned to the mechanisms based on temperature measurements. As summarized by the table, the aims of the present paper are partially matched by the current literature. The table also includes another important topic: the applicability of the algorithm in a real context. This pertains computational cost and ease of implementation and it is archived by algorithms requiring simple mathematical operations (such as summations, multiplications); a topical example is the one of [17] or the application of linear regression [19]. The papers highlighted with a • mark on applicability hardly match the other requirements. The present approach may require a computational expensive training phase. We consider such a training phase the necessary step to achieve a good compromise between performance and applicability when adaptation to unknown noises, temperature and asymmetry compensations are required. Elaborating a countermeasure to those factors on the basis of samples of the system in a single algorithm is the topical issue addressed in this work. For this reason the paper is presented in the table (last row) with all • marks, except for the ◦ mark on applicability in virtue of the computational complexity of training. Similar considerations may hold for the use of the least square method of [21] or for the multi-model Kalman filter in [14] and PF in [10]. Another point of strength of the present work is the performance metric used (the 99.9 percentile of the synchronization error), which is even more stringent than the 90% and 92% confidence intervals of average absolute error of [31] and [21], respectively.

3. Problem formulation

3.1. The two-way timing message exchange

We are mostly considering WSN and WiFi networks by focusing on pairwise synchronization (synchronization between a pair of neighboring nodes) rather than network-wide synchronization (hierarchical pairwise synchronization) [10]; the network-wide model is a generalization of the pairwise model as outlined in [16]. We take the *two-way timing message exchange* mechanism as a reference. This basic synchronization scheme is typical of many *sender-receiver* CSPs, such as, e.g., the Timing-sync Protocol for Sensor Networks (TPSN) [32] and the *timing measurement* mechanism defined in the recent IEEE 802.11-2012 specification of WiFi [33]. We consider two nodes, called *sender* and *receiver*, which periodically take and exchange timestamps of their internal clocks. The sender is the one starting the exchange that consists of 3 packets. On sending and reception of the first 2 packets, 4 timestamps are acquired, denoted by t_1, t_2, t_3, t_4 (see Fig. 1, in which the timestamps are reported in **bold**). t_1 is the sending time of a synchronization packet from the sender to the receiver (under the notion of time of the sender). t_2 is the time of the receiving of the packet at the receiver and t_3 is the time of the sending of the response synchronization packet from the receiver to the sender. Both t_2 and t_3 are defined under the notion of time of the receiver. The last packet (from the receiver to the sender) includes the values of t_2 and t_3. Finally, t_4 is the time of the receiving of the first response packet (under the notion of time of the sender). The involved delays evidenced by Fig. 1 will be detailed later. At the end of the exchange, a measure of the synchronization state (offset, skew, together with the delay) is obtained. In turn, the state estimation is updated from the collected measurements at the end of each exchange; typically a set of K message exchanges is exploited for state estimation [17, 16]. The sender makes use of the state estimation to synchronize its clock to that of the receiver. A software layer, named for the first time *virtual clock* in [34], converts the sender time into the receiver time. A virtual clock is essential for devices that cannot adjust the clock register at runtime. Synchronization is performed at each discrete time instant $k, k+1, ...$; let τ be the size of those discrete time steps. We assume each message exchange starts and ends in $[k, k+1], \forall k$.

We now enter in the details of the state and measurement models.

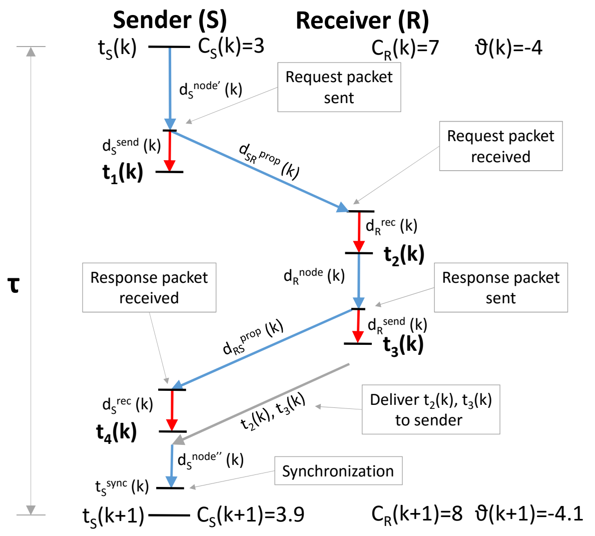


Figure 1. Timestamps, messages exchange, and involved delays.

3.2. State equations

The clock registers of the sender and the receiver (denoted by $C_S(k)$ and $C_R(k)$, respectively and reported in Fig. 1) differ of the offset quantity $\theta(k)$.

$$C_S(k) = C_R(k) + \theta(k) \tag{1}$$

A typical clock model can be represented by the following equations in the discrete domain:

$$\begin{aligned} \theta(k) &= \theta(k-1) + \gamma(k-1) \cdot \tau + \omega_\theta(k-1) && (2) \\ \gamma(k) &= \gamma(k-1) + \omega_\gamma(k-1) && (3) \\ d(k) &= d(k-1) && (4) \end{aligned}$$

where $\gamma(k)$ represents the skew and $d(k)$ the delay component, which is assumed stationary over time, but whose measurements are affected by noise as detailed later in subsection 3.5. The stationarity assumption is motivated in the same section as well.

3.3. State noise

The ω_θ and ω_γ quantities represent the noises affecting θ and γ, respectively, whose distributions are usually modeled as Gaussian type; the corresponding standard deviations are denoted by σ_{ω_θ} and σ_{ω_γ}, respectively.

Here and in [15] we include in ω_γ an additional component,

$$\omega_{\gamma^T}(t, \cdot) = \omega_{\gamma^T}(t, tc, p, T_E^{High}, T_E^{Low}) \tag{5}$$

to mimic the temperature effect of a XO periodically moved every p [s] between two environments with temperatures T_E^{High} [$°C$] and T_E^{Low} [$°C$]; tc is the XO *thermal time constant*, which represents the thermal inertia of the XO and of its case.

The *Newton's law of cooling*

$$T = T_E + (T_{XO} - T_E) \cdot e^{-\frac{1}{tc}\Delta t} \tag{6}$$

models the evolution over time of the temperature of an object with initial temperature T_{XO} which is placed in an environment characterized by a new temperature T_E. The periodic movement between the two environments with different temperatures has been performed in simulation every p, and in such instants T_{XO} is set equal to the current temperature T of the XO and T_E is set equal to T_E^{High} or T_E^{Low}.

The XO is modeled as an AT-cut quartz, the more common in real devices and the one that suffers more the temperature effects. Given the temperature T, the frequency variation from the nominal one can be properly approximated as

$$\frac{\Delta f}{f} = a \cdot (T - T_0) + b \cdot (T - T_0)^2 + c \cdot (T - T_0)^3 \qquad (7)$$

where T_0 is the reference temperature of the XO and a, b, c are three constants modeling the XO. This paper makes use of parameters directly derived from a real AT-cut quartz [35]: $T_0 = 25\,°\mathrm{C}$, $a = 0.0$, $b = 0.4 \cdot 10^{-9}$ and $c = 109.5 \cdot 10^{-12}$.

The quantity $\omega_{\gamma^T}(t, \cdot)$, which corresponds to the frequency variation $\frac{\Delta f}{f}$ at a given time t, can be easily derived by substituting (6) in (7).

We noticed that the temperature variations have a predominant effect on the synchronization error rather than other variations of the parameters in (7) (a, b, c, T_0). For this reason, the performance evaluation concentrates the attention on large temperature variations under a realistic XO model [35], with fixed parameters.

The inherent probability distribution of $\omega_{\gamma^T}(t, \cdot)$ is not Gaussian; it is actually a multi-modal distribution, with significant asymmetry among the peaks.

3.4. Measurement equations

Let $t_1(k), t_2(k), t_3(k), t_4(k)$ the timestamps in $[k-1, k]$. A sample of the delay at time k, $\check{d}(k)$, is derived as follows:

$$\check{d}(k) = \frac{\big(1 - \gamma(k)\big) \cdot \big(t_4(k) - t_1(k)\big) - \big(t_3(k) - t_2(k)\big)}{2} \qquad (8)$$

The equation, used by CSPs to compute the propagation delay, clearly outlines the non-linearity of the model (between delay and skew) and the indirect impact of the temperature noise on delay estimation through the skew.

For every exchange of synchronization messages, two samples of θ, namely $\check{\theta}_{SR}$ and $\check{\theta}_{RS}$ can be computed by:

$$\begin{aligned} \check{\theta}_{SR}(k) &= t_1(k) - (t_2(k) - d(k)) \\ \check{\theta}_{RS}(k) &= t_4(k) - (t_3(k) + d(k)) \end{aligned} \qquad (9)$$

The use of both equations to compute the offset, peculiar of the two-way timing message exchange mechanism, allows a better estimation of the offset.

A sample of the skew, $\check{\gamma}(k) = 1 - m$, is derived by calculating the slope m of the line interconnecting the two points $(t_1(k), t_2(k) - d(k))$ and $(t_4(k), t_3(k) + d(k))$, placed in the sender-receiver space. The slope represents the ratio between the receiver and the sender oscillation periods of the XO. More specifically, the differences t_4-t_1 and t_3-t_2+$2d$ lie on the sender and receiver time-lines, respectively, as outlined in Fig. 2.

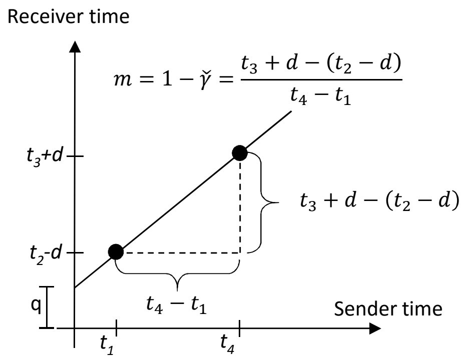


Figure 2. Skew sample $\check{\gamma}$ from the slope m of the interpolation of t_1, t_2, t_3, t_4.

$$\check{\gamma}(k) = 1 - \frac{t_3(k) + d(k) - (t_2(k) - d(k))}{t_4(k) - t_1(k)} \qquad (10)$$

An intuitive example of the application of (8)-(10) is provided in Appendix A. Operatively, the measurement equations may be simplified as follows. By assuming delay stationarity over each observation period $[k-K, ..., k], k = 1, 2, ...$, a delay estimation at time k, $\hat{d}(k)$, may be derived from the arithmetic average of $\check{d}(k-K), ..., \check{d}(k)$ and by simplifying $\gamma(k)$ in (8) with $\hat{\gamma}(k-1)$ (i.e., with the previous estimation of γ; $\hat{\gamma}(1) = 1$). The $\hat{d}(k)$ quantity may be then substituted in (9) and (10). This kind of calculation is however not necessary for the proposed estimation schemes as outlined in subsection 5.1. The stationarity assumption on delay means that nodes mobility takes place over time horizons larger than K (i.e., the size of the observation period). In case of mobility over shorter ranges, the state equation (4) should be updated with an appropriate noise variable or with an additional function mapping $d(k-1)$ in $d(k)$.

3.5. Measurement noise

The noise affecting the equations above is due to the randomness of the values of t_1, t_2, t_3, t_4, which derives from the chain of delays evidenced in Fig. 1 and defined as follows (the index k is not mentioned for the sake of clarity).

By following the clock register C_S, at time t_S (top-left of the figure), the sender schedules the sending of a synchronization packet. After a $d_s^{node'}$ delay (corresponding to the operational times of the local operating system, the Media Access Control (MAC) and the transceiver) the packet is actually sent, and its sending time t_1 can be recorded by the sender node after a d_s^{send} delay (corresponding to the operational time for handling the time-stamping procedure triggered by the network adapter[4]).

[4]In order to improve precision of software timestamping, the timestamping operation is scheduled by the first instruction of the inherent Interrupt Service Routine (ISR). Moreover, the network adapter is sometimes assumed to be able to insert timestamps on-the-fly, just before the packet is transferred over the air. In this case, the third packet exploited by the protocol to deliver t_2 and t_3 to the sender is no longer required. Some other adapters acquire the timestamps in hardware, i.e., at MAC level, thus allowing the reduction of delay oscillations even more.

The packet arrives at the receiver after d^{prop}_{SR}, i.e., the medium propagation delay from the sender to the receiver.

Analogous delay quantities are defined for the receiver when setting the values of t_2 and t_3 (namely, d^{rec}_{R} and d^{send}_{R}), for the sender again when setting the value of t_4 (i.e., d^{rec}_{S}), and for the medium propagation delay d^{prop}_{RS} for packets sent in the opposite direction. The last packet is sent from the receiver to the sender (bottom right in the figure); it contains the values of t_2 and t_3 needed for a clock correction step. It is sent after the first response packet, which triggers the t_4 computation at the sender. All t_1, t_2, t_3, t_4 are involved in the synchronization update through (8)-(10). The clock update occurs at time t^{sync}_{S} after a delay $d^{node''}_{s}$ (bottom left in the figure), that includes the inherent estimation procedure and other in-node overheads.

It is important to remark that the traffic generated by other interfering nodes in the wireless network does not influence the precision and the accuracy of the timestamps. In practice, timestamps t_1 and t_3 are recorded after the transmitting node has already started the transmission on the ether, and after a possible wait due to carrier sense. As a consequence, t_1 and t_3 do not include the error due to the indeterminism of the access schema used by the MAC layer of the wireless communication protocol. On the receiving side, timestamps t_2 and t_4 are acquired as quickly as possible after the arrival of the synchronization message, and the only possible effect of interfering traffic and disturbs is the loss of some messages. The effect of losses for a technique based on a 1-st order regression spline was analyzed in a real implementation in [6]. Results reveal that the impact of losses on synchronization quality is negligible when 1-st order regression spline is computed on at least 70% of the expected points, and they do not worsen considerably when the number of losses further increases. It can safely assumed that the same applies to the approach based on a neural network proposed in this paper, because it uses a 1-st order regression spline for features extraction. A more detailed analysis of this aspect, and how to further improve accuracy for the proposed technique, is left open for future research.

4. Optimal state estimation

In principle, the problem consists of defining the optimal filter for state estimation at each time k, on the basis of measurements collected up to time k [15]. The optimal filter is a function[5] $\boldsymbol{\nu}^o(\cdot)$ that maps the measurements into the estimates at each time k. Let $\boldsymbol{x}(k) = \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{\xi}(k-1))$ be the state equation in compact form from (2), (3) and (4) with $\boldsymbol{x}(k) = [\theta(k), \gamma(k), d(k)]$, $\boldsymbol{\xi}(k)$ being the vector of state noises and $\boldsymbol{y}(k) = \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{\eta}(k))$ the measurement equation from (8), (9) and (10) with $\boldsymbol{y}(k) = [\check{\theta}_{SR}(k), \check{\theta}_{RS}(k), \check{\gamma}(k), \check{d}(k)]$, $\boldsymbol{\eta}$ being the vector of measurement noises, respectively. The optimal estimation law $\boldsymbol{\nu}^o_k(\cdot) = \boldsymbol{\nu}^o_k(\boldsymbol{I}_k)$ minimizes the following functional cost:

$$\boldsymbol{\nu}^o_k(\boldsymbol{I}_k) = arg \min_{\boldsymbol{\nu}_k(\boldsymbol{I}_k)} \mathop{\mathrm{E}}_{\boldsymbol{x}(k)} \{\boldsymbol{h}(\boldsymbol{x}(k) - \boldsymbol{\nu}_k(\boldsymbol{I}_k))|\boldsymbol{I}_k\}, \forall\, \boldsymbol{I}_k \tag{11}$$

$\boldsymbol{I}_k$ being the information vector collecting all the measurements from the beginning $\boldsymbol{I}_k = [\boldsymbol{y}(0), ..., \boldsymbol{y}(k)]$ and $\boldsymbol{h}(\cdot)$ being a Bayesian risk function[6]. The optimal filter cannot be derived in closed-form as in the Kalman filter owing to the non-linearity of (8) and to the temperature noise which is not Gaussian. Here we resort to an approximating technique, as later outlined in Section 6.

5. Splines

Before addressing the approximation of the optimal filter, a basic heuristics is defined. If Fig. 2 includes the collection of K sets composed of 4 timestamps (t_1, t_2, t_3, t_4), the trend of the current asynchronism may be derived by interpolating the corresponding $2K$ points in the sender-receiver space, thus deriving a heuristic estimation of γ and θ. This is the underlying idea of the splines. The following information vector is defined:

$$\boldsymbol{I}^s_k = \left[\varrho^{t_2,t_1}_{k-K}, ..., \varrho^{t_2,t_1}_{k}, \varrho^{t_3,t_4}_{k-K}, ..., \varrho^{t_3,t_4}_{k}\right] \tag{12}$$

with:

$$\varrho^{t_2,t_1}_{k-j} = \left[t_2(k-j), t_1(k-j)\right]; \quad j = 0, ..., K. \tag{13}$$

$$\varrho^{t_3,t_4}_{k-j} = \left[t_3(k-j), t_4(k-j)\right]; \quad j = 0, ..., K. \tag{14}$$

where ϱ^{t_2,t_1}_{h} and ϱ^{t_3,t_4}_{h} pertain the timestamps of Fig. 2 at time $h = k-j$ and $\boldsymbol{I}^s_k$ pertains the collection of timestamps in $[k-K, ..., k]$.

The i-th order spline (denoted by S_i) is derived by interpolating, with the i-th order, the set of points in $\boldsymbol{I}^s_k$ by means of the Ordinary Least Squares method [37]. As intuitively summarized by Fig. 2, a skew estimation is derived from the slope of S_1 and the offset estimation from putting in the spline equation the current value of time of the sender. Quadratic and cubic splines help chase the non-linearity of the offset correction. Orders higher than 3 have been disregarded to avoid overfitting, to which the splines are more sensitive with more noise and large K. At the sender, the synchronization step consists of directly putting the current notion of time, C_S, into the spline equation, thus deriving the current notion of time at the receiver C_R. Because the slope of a 1-st order spline is an estimation of $\gamma(k)$, S_1 perfectly compensate all the stationary effects on the skew (e.g., the real oscillation frequency of the XO which differs from the nominal one), and all the effects on frequency with a periodicity much greater than τ, such as aging. Effects with periodicity slightly greater than τ are compensated by S_1 only on average, but they can be addressed by the technique based on a neural approximation described in the next section. Examples are temperature variations or vibration with a periodicity greater than τ. Periods lower than τ are typically disregarded. A remedy is the reduction of τ according to the specific working conditions.

[5] *Filter*, *estimation law* or *estimation function* are typically used as synonyms.

[6] $\boldsymbol{h}(z)$ is a Bayesian risk function if the following are met: $\boldsymbol{h}(z)$ is not negative, it is symmetric, i.e., $\boldsymbol{h}(z) = \boldsymbol{h}(-z)$ and it is not decreasing with increasing positive z; in the scalar case, examples are: $h(z) = z^2$ and $h(z) = |z|$. Such a risk function is used in statistical decision theory as a measure of the difference between the estimation and the true value (see, e.g., subsection 1.2.1 of [36]).

5.1. Delay correction

Let d represents the knowledge of the delay, derived in any way; it may be either the propagation delay, including in-node overheads or not, or the delay estimation outlined in subsection 3.4. This knowledge reduces the noise on the timestamps t_2, t_3, if we apply a correction to replace them with: t_2-d and t_3+d. In principle, this may drive a better state estimation, but it reveals to be useless for the splines. If a sufficient number of samples is taken, the regression schemes lead to identical curves, independently to the application of the correction. Intuitively, this is due to the averaging operation operated by regression while capturing the trend of the timestamps. We empirically validated this property for all the splines considered; a formal demonstration for S_1 is provided in Appendix B.

6. Neural approximation

A further generalization in the direction of tracking the non-linearity and non-stationarity of the involved processes is now addressed. The focus is on the offset, thus disregarding skew and delay estimation. Similarly to the splines, the offset estimation allows the direct implementation of the synchronization step by applying, as described in the previous Section 5, the approximating function (based either on splines or on a neural network as derived here). A standard Neural Network (NN) training is formulated as follows. A new information vector is defined:

$$\boldsymbol{I}_k^{NN} = [\delta_{k-2K}, ..., \delta_k] \tag{15}$$

δ_k being the distance between each ϱ_h in $\boldsymbol{I}_k^s$, as defined in (12) and with $h = k-2K, ..., k$, and the first order spline (S_1) interpolating the set of points in $\boldsymbol{I}_k^s$. The corresponding training set is stated with $k = 1, ..., N$ samples of $\boldsymbol{I}_k^{NN}$ and the relative target ε_k, i.e., it is a set of N tuples in the form $\langle \boldsymbol{I}_k^{NN}, \varepsilon_k \rangle$. The value ε_k being the error of S_1, evaluated for a given time k in the sender timescale, in predicting the time in the receiver space. Given the S_1 spline obtained for $\boldsymbol{I}_k^s$, $S_1^k(k) = m^k \cdot k + q^k$, the target can be computed as $\varepsilon_k = C_R(k) - S_1^k(k)$.

The optimal weights assignment $\boldsymbol{w}^o$ is derived so that:

$$\boldsymbol{w}^o = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}); J(\boldsymbol{w}) = \sum_{k=1}^{N} [\varepsilon_k - \hat{\nu}_k(\boldsymbol{I}_k^{NN}, \boldsymbol{w})]^2 \tag{16}$$

Problem (16) can be solved by applying standard non-linear optimization techniques. In particular, the proposed NN, which has been parameterized through an extensive series of experiments, has three layers: the first with $2K$ input nodes, the hidden layer with 10 nodes characterized by hyperbolic tangent activation functions, and one linear output node in the last layer. It was trained using 8 iterations (epochs) of the classical *back-propagation* training algorithm. During the 8 epochs, at each iteration on the training set the *learning rate* was decreased linearly between 0.001 and 0.00001. The *momentum* was set to 0.01. In the test phase, the output of the NN is added to the estimation of the receiver time, which is performed by computing S_1 on a time expressed in the sender timescale.

The sequence of approximation steps from the optimal filter to this scheme are the same referenced in [15]. Here, we stress the fact that (16) is solved with respect to samples coming from non-stationary noises. This consequently leads to the adaptation of the approach to variable system conditions. In this respect, differently from [14, 19], no online adaptation of the algorithm is required, and, differently from PF [10], the computational effort of the approach resides in computing $\boldsymbol{w}^o$ offline.

7. Implementation issues

7.1. Deployment of the neural estimator

Three steps are crucial for the deployment of the neural estimator: acquiring the synchronization error ε in (16), building a database for training and applying the training procedure (i.e., solving (16)). The first step can be performed through either specialized devices [38] or by referring to a reference signal (e.g., a periodical actuation function [6] or the deterministic expected time of reception in TDMA [8]). An external device is in charge as well to perform the remaining two steps.

Those steps may be hardly applied in multi-hop networks if they should be repeated for each node in the network. For this reason, we derive a method to join the replication of the steps into a single procedure in the multi-hop context (i.e., one database and one training).

7.2. Computational cost

One main aim of [17] is to derive a simple synchronization technique with less computational complexity than Linear Programming (LP) or other traditional approaches [10]. This is crucial for WSNs in which energy is a scarce resource. In the LP case, for example, an optimization problem is formulated on skew and offset. The problem can be solved by traditional optimization algorithms, like the simplex one. The simplex method is efficient in practice, even though it has exponential worst-case complexity. The computational issue may become critical in the synchronization context because the number of LP constraints scales up linearly in the size of the information window collecting the history of the timestamps (see, e.g., (21) of [10]). Despite the splines advocate the adoption of the Ordinary Least Squares method, for which similar considerations may be outlined, their computational complexity is low, in particular for S_1. In the bivariate case (the regression is applied on the plane reported in Fig. 2), each new sample contributes to the updating of the S_1 parameters through trivial operations, such as the update of the sum of products on previous samples (see, e.g., [19] and [39]). The *NN* experiences a low computational complexity as well because it depends on the collection of S_1 estimations. After building $\boldsymbol{I}^{NN}$ on S_1, the remaining *NN* operations consist of computing the *NN* output through a cascade of summations and multiplications involving the neural units, typically represented by hyperbolic tangent or sigmoidal functions. The *NN* used in the performance evaluation has been implemented on an Atmel ATmega328P microcontroller running at 16 MHz and tested with a *NN* consisting of 15 inputs, 10 hyperbolic tangent hidden units and one linear output. Such a microcontroller is

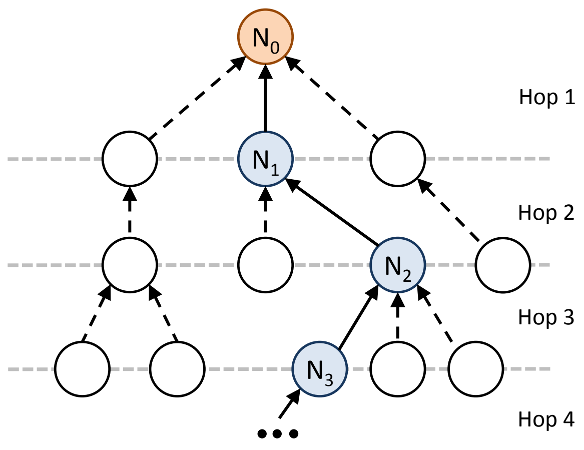


Figure 3. Example of a multi-hop network.

of common use in the WSN context. The registered mean execution time of each iteration involving both features extraction (from S_1) and the computation of the output (after training) was 4.932 ms [15]. The *NN* computation also scales linearly with respect to K. Roughly speaking, this corroborates the adoption of the *NN* with $\tau \geq 100$ ms, τ being the size of the synchronization time steps. An accurate calibration of K and of the other *NN* parameters (the number of hidden units, in particular) deserves further attention if smaller synchronization steps are required.

The computational cost of the *NN* train phase is higher than the one for test. Actually, the duration of training is not a limitation because it is executed offline. Unless not differently specified, all NNs have been trained with 100000 samples, which in real systems must be acquired at runtime through measurements. As a consequence, the time needed to acquire the train database, with $\tau = 1$ s, is about 27 hours. The offline training time, measured on a PC equipped with an Intel Core i7-3770 CPU running at 3.4 GHz, with a not optimized software and with $K = 60$ (i.e., 120 inputs) is about 10 minutes. This time can be reduced of at least one order of magnitude with software optimization or by exploiting GPUs [40].

8. Multi-hop analysis

In a number of operating conditions, usually in large networks, nodes communicate through intermediate devices. In the viewpoint of synchronization, the nodes are hierarchically ordered in a *tree* topology [10, 19], where at the root of the tree lies the *reference clock* (i.e., the time source of the network tree). The first layer nodes synchronize directly with the reference clock. A second layer node synchronizes with the first layer. The same applies for the subsequent layers. Each layer suffers of a worst synchronization quality as soon as the distance from the root increases.

Given a node, only one path exists between the node and the root of the tree. Synchronization can be separately analyzed along each path of the tree. We focus on the example reported in Fig. 3. The reference clock, node N_0, has only the *master* role, i.e., it is the time source to which all other nodes must synchronize. A node of the first layer, N_1 has a 1-hop distance from N_0, and it acts as a *slave* node with respect to N_0 and as a master node with respect to the nodes lying in the second layer, for example N_2. The node N_2, that like N_1 covers both the master and slave roles, has a 2-hop distance with N_0. The node N_2 is synchronized with N_1, which in turn is synchronized with N_0. Basically, timestamps obtained on timing messages by node N_1, when N_2 synchronizes with N_1, are obtained with the view of N_1 of the reference time held by N_0. In other words, N_1 exploits the most recent estimation of the parameters of the virtual clock to convert the timestamps obtained with its local clock to the reference time held by N_0. Since the virtual clock makes errors in this conversion, a node belonging to a hop level greater than 1 synchronizes its clock to an incorrect clock source. We will refer to such a kind of corruption as *hop error*. The hop error increases with the number of hops and does not take place in N_1. Leaf nodes have only the slave role and are the ones with the biggest hop error.

Three methods based on *NN*, namely NN_{link}, NN_{gen_A} and NN_{gen_B}, are defined in this context.

8.1. A distinct NN for every link

The NN_{link} model implies that the usual NN is trained link-by-link, i.e., by repeating a training phase for every communication link between two adjacent nodes of the network. From the second hop onwards, the neural network is trained with timestamp data derived from a node which is synchronized through a NN with the node of the previous hop. This method come out with a set of specialized NNs, able to cope with the specific experimental conditions regarding each couples of nodes. Unfortunately, this may result in a high number of database acquisitions and neural trains. Moreover, to acquire the training database for a specific hop, the NN of the previous hops must already be trained. This iterative approach leads to a complex acquisition process of the training databases and waste of time for the system setup. As a matter of fact, it is hardly applicable in real situations, except when the number of nodes is reasonably limited.

8.2. Generalized NN: temperature compensation

The limit of having a specific NN for every link of the network can be circumvented by training with respect to different temperature patterns. The accuracy of this kind of "generalized" NN is usually lower than the one of a NN trained with a database coherent with the test conditions (see, e.g., [15]). We firstly define a NN trained over a set of temperature conditions, in-node delays, but by disregarding the hop error: NN_{gen_A}. This means the training database can be easily obtained by connecting in separate, but not consecutive, experiments all the nodes with the master.

8.3. Generalized NN: temperature and hop error compensation

Under the NN_{gen_B} model, a single neural network is trained under an iterative (link-by-link) approach, as done in the NN_{link} case. The difference relies on the superimposition of pairs of temperature and hop error conditions, which differ from the combinations checked in the test phase.

In particular, the function $\omega_{\gamma^T}(t, \cdot)$ modeling the effects of temperature variation on the skew has been chosen to be different in the train and test phases. To this purpose, the temperature models evaluated in the test phase for the last and penultimate nodes drive the collection of training data at the first and second nodes, respectively, and so on. As an example, in a network with 5 hops, $\omega^1_{\gamma^T}$, $\omega^2_{\gamma^T}$, ..., $\omega^5_{\gamma^T}$ are the temperature models used in the test phase for nodes N_1, N_2,...N_5, respectively. The temperature models associated to the same nodes (i.e., N_1, N_2,...N_5) in the train phase are $\omega^5_{\gamma^T}$, $\omega^4_{\gamma^T}$, ..., $\omega^1_{\gamma^T}$. The resulting database contains a mix of hop errors and temperatures over which the NN learns the clock correction, independently to the knowledge of its position in the tree branch. More sophisticated methods for the generation and synthesis of the training database are left open for future research.

9. Experiments setting

The following simulation parameters are defined for the performance evaluation.

Observation horizon. The value K is the number of sets of timestamps $\{t_1, t_2, t_3, t_4\}$ used to decide the synchronization correction. K is used as a variable parameter in the results (it appears in the x-axis of all the figures) in order to emphasize how the techniques may be sensitive to it. As signal processing techniques estimate stationary states, they achieve optimal performance only for a strict range of K in non-stationary conditions, such as in the presence of temperature variations.

Synchronization period. In the proposed simulation model, the sender node starts a synchronization step in a cyclic fashion, and each exchange is triggered with a period τ fixed to 1.0 s. Periods greater than 1.0 s could be useful in contexts where power consumption is a main target, such as in WSNs. The effects of the parameter τ on synchronization quality has been analyzed in a specific experimental campaign in subsection 10.3.

Performance metric. The 99.9 percentile of the synchronization errors is the performance metric. It is denoted by $p99.9$ and represents the 99.9 percentile of the absolute difference between the reference time (i.e., the time at the receiver node) and the estimated time by the sender at the end of each timestamps exchange, as outlined in Fig. 1. All the performed simulations (under a fixed K) contain 100000 samples of the dynamic system (2)-(4) whose evolution follow the temperature and delay models presented below.

The average error is disregarded because it represents only the systematic part of the error (i.e., the *accuracy*), but it does not provide any information about the *precision* of synchronization [41]. Two nodes may be synchronized, on average, while still experiencing large synchronization errors; $p99.9$ helps capture a threshold limit of those errors (in the 99.9% of the cases).

Temperature. The ω_θ and ω_γ components of equations (2) and (3) have variances $\sigma^2_\theta = 10^{-17}$ s^2 and $\sigma^2_\gamma = 10^{-19}$, respectively. As far as the temperature is considered, two models have been taken into account. The $\omega^{high}_{\gamma^T}(t, \cdot)$ model represents a fast

Table 2. Means and standard deviations of in-node latencies

Scenario		sw^{WiFi}	hw^{WiFi}	sw^{WSN}	hw^{WSN}
Type		PC-WiFi	PC-WiFi	Mica2	TelosB
Latency		(μs)	(μs)	(μs)	(μs)
d^{send}_S	μ^{send}_S	5.4	1.31	259.057	0.408
	σ^{send}_S	0.310	0.046	1.291	0.0157
d^{rec}_S	μ^{rec}_S	7.23	8.9	346.849	2.769
	σ^{rec}_S	0.580	0.110	2.415	0.0374
d^{send}_R	μ^{send}_R	$n_\mu \cdot 5.4$	$n_\mu \cdot 1.31$	$n_\mu \cdot 259.057$	$n_\mu \cdot 0.408$
	σ^{send}_R	$n_\sigma \cdot 0.310$	$n_\sigma \cdot 0.046$	$n_\sigma \cdot 1.291$	$n_\sigma \cdot 0.0157$
d^{rec}_R	μ^{rec}_R	$n_\mu \cdot 7.23$	$n_\mu \cdot 8.9$	$n_\mu \cdot 346.849$	$n_\mu \cdot 2.769$
	σ^{rec}_R	$n_\sigma \cdot 0.580$	$n_\sigma \cdot 0.110$	$n_\sigma \cdot 2.415$	$n_\sigma \cdot 0.0374$

temperature variation of the XO ($t = 600$ s) in a wide range of temperatures ($T^{Low}_E = -10\,^\circ$C and $T^{High}_E = 40\,^\circ$C); while the $\omega^{norm}_{\gamma^T}(t, \cdot)$ model is characterized by slower temperature variation ($t = 1200$ s) than $\omega^{high}_{\gamma^T}(t, \cdot)$, in a narrower range of temperature with extremes $T^{Low}_E = 10\,^\circ$C and $T^{High}_E = 35\,^\circ$C. For both temperature models $tc = 60$ s. The models are applicable to mobile nodes in reality. An example may be an automatic forklift that enters into and exits from an industrial oven. Outdoor exposure is applicable as well [42, 21].

Delay. We consider two effects on delay: the propagation over the channel, d^{prop}_{SR} and d^{prop}_{RS}, and the *in-node* delays (d^{send}_S, d^{rec}_S, d^{send}_R and d^{rec}_R), as defined in the following. The values $d^{prop}_{SR} = 150$ ns and $d^{prop}_{RS} = 150$ ns have been used in all the simulated scenarios. The value of 150 ns was chosen because it represents a reasonable distance of about 50 meters between wireless nodes. In fact, an electromagnetic signal which a speed of $\sim 3 \cdot 10^8$ m/s takes 166 ns to cover 50 m. The propagation delay is also considered as exponentially distributed in the last experiments of subsection 10.5. The other delays of Fig. 1, unless otherwise specified, have been set to 0 (i.e., $d^{node'}_S = d^{node''}_S = d^{node}_R = 0$).

Comparison with [17]. As a performance comparison, in the method summarized by the *CH* acronym, formulas (10) and (11) of [17] have been chosen for the estimation of the skew and offset, respectively. In the proposed experimental setups, they provide the best results with respect to the other variants presented in the same paper.

9.1. In-node delays

In-node delays represent the latency between the sending or reception times of the synchronization packet and when the timestamp is actually obtained. In absence of nodes' mobility or other structural changes of the environment, which may cause variations of the fading affecting the channel, the propagation delay is deterministic and the prevailing effect on synchronization is due to in-node delays.

The setting of the noise on in-node delay is now detailed. In-node delays reported in Fig. 1 are more concisely represented with: d^{send}_ρ and d^{rec}_ρ, where ρ represents the node role, $\rho = S$ for the *sender* node and $\rho = R$ for the *receiver* node, respectively.

In [43], in-node delays have been analyzed for IEEE 802.11 WiFi devices and results are reported for hardware and software timestamps. The resulting delay model is summarized in the first two columns of Table 2. The two latencies d_ρ^{send} and d_ρ^{rec} are not symmetric, and sender and receiver have very different in-node delays values in terms of both mean and standard deviation (for instance sw^{WiFi} and hw^{WiFi} in Table 2).

The presented setting for software time-stamping (condition sw^{WiFi} reported as first column of Table 2), is referred to a system with low interfering loads (i.e., CPUs often in the *IDLE* state, low interrupts rate, etc.). With hardware timestamps, hw^{WiFi} condition, the standard deviations σ_ρ^{send} and σ_ρ^{rec} are quite small.

The distribution of d_ρ^{send} and d_ρ^{rec} [43] may have various shapes depending on the nodes hardware, operating system and internal load. A good approximation is however the normal distribution: $d_\rho^{send} = \mathcal{N}(\mu_\rho^{send}, \sigma_\rho^{2\ send})$ and $d_\rho^{rec} = \mathcal{N}(\mu_\rho^{rec}, \sigma_\rho^{2\ rec})$. The multipliers n_μ and n_σ in Table 2 are used in subsection 10.4 in order to set variable asymmetry conditions on the delays d_ρ^{send} and d_ρ^{rec} of the receiver node. The quantities $d_{\rho,i}^{send} - \mu_\rho^{send}$ and $d_{\rho,i}^{rec} - \mu_\rho^{rec}$ are usually known as *jitter*.

As far as WSNs are considered, some papers [19, 44, 45] have experimentally evaluated and analyzed the distributions of d_{SR}^{path} and d_{RS}^{path}, where:

$$\begin{aligned} d_{SR}^{path} &= d_{SR}^{prop} + d_R^{rec} - d_S^{send} \\ d_{RS}^{path} &= d_{RS}^{prop} + d_S^{rec} - d_R^{send} \end{aligned}$$

represent the measured path delays between sender and receiver, and vice versa. At the best of authors knowledge, a separate analysis for in-node delays is not currently available. In [44], the transmission latency of a message between WSN nodes of different type have been evaluated. In particular, for a Mica2 WSN node the reported delay is 605.906 μs, with a standard deviation equal to 2.738 μs.

In order to have a coherent model for WSN, we split the values reported in [44] between the d_ρ^{send} and d_ρ^{rec} contributions, with the constraint of maintaining the ratios $\frac{\mu_\rho^{send}}{\mu_\rho^{rec}}$ and $\frac{\sigma_\rho^{2\ send}}{\sigma_\rho^{2\ rec}}$ experienced for sw^{WiFi}. The previously described process was used to obtain the sw^{WSN} condition (third column of Table 2).

The same splitting procedure, between the sending and the receiving components of the delay, has been used for the hw^{WSN} condition. In this case, data about delays have been obtained from [19], and the constraint regarding the two ratios are coherently referred to a hardware condition, i.e., hw^{WiFi}. In [19], the experimental setup is composed of TelosB WSN motes, and the measured delay of a transmission between two nodes of this type — including in-node delays but excluding the timestamps quantization errors — has a mean value and a standard deviation of 3.177 μs and 40.56 ns, respectively. An important distinguish characteristic of the hw^{WSN} scenario is that the timestamps resolution is $\sim 31\,\mu$s, because nodes local clock has a frequency of 32768 Hz. As a result, even if the precision of the timestamp is very high (i.e., the standard deviation is below 50 ns), the use of a great number of timestamps is mandatory to mitigate the effects related to their low resolution.

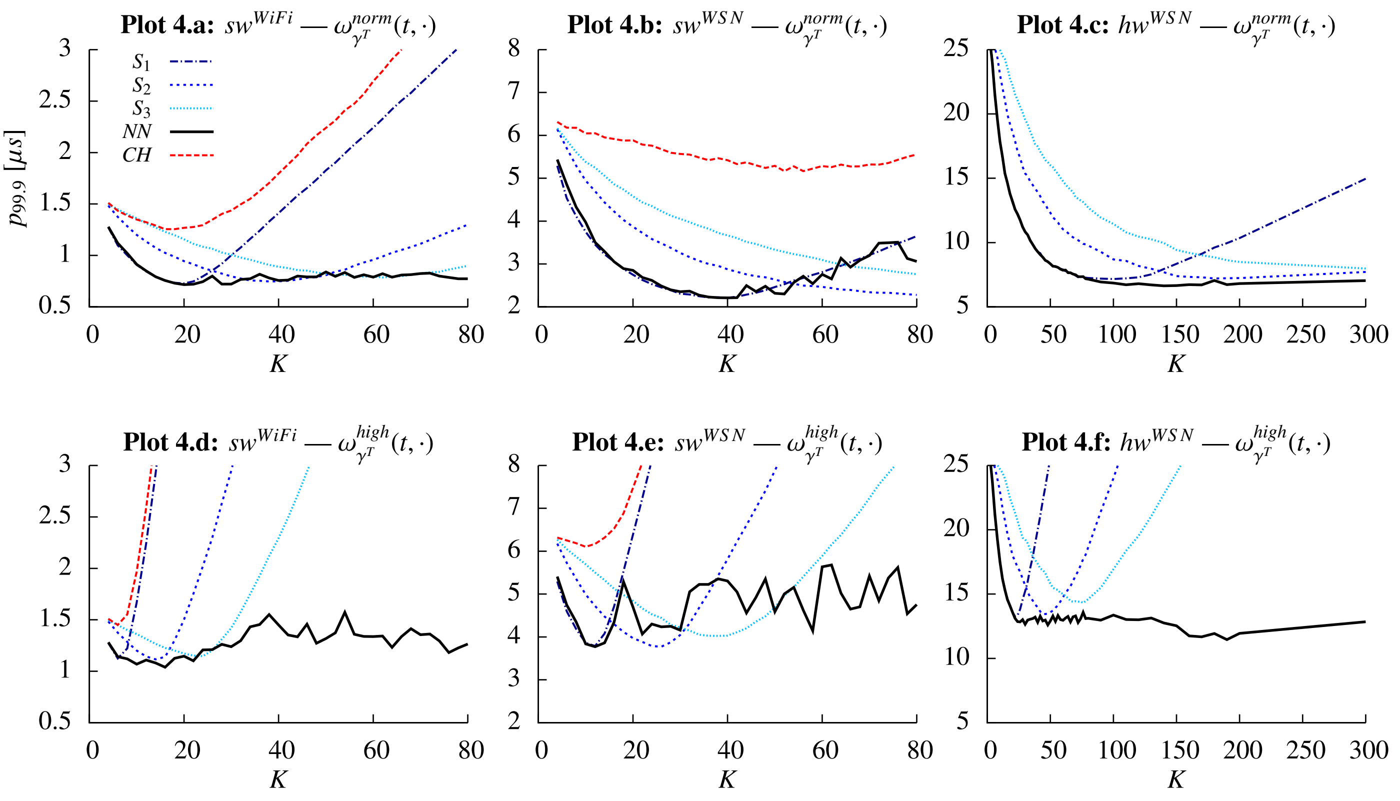
The use of more than one timestamp to drive estimation (i.e., $K > 1$) helps mitigate in-node jitter or quantization errors for all the conditions, with the only exception of hw^{WiFi}, whose timestamps are characterized by a higher precision. Under hw^{WiFi} conditions, small values of K, e.g., $K = 2$, are sufficient for synchronization with S_1. Under peculiar circumstances, e.g., with high values of τ, or communication losses between sender and receiver, further adjustments may be needed; this has been left open for future research.

10. Performance evaluation

10.1. Simulation environment

Typical network simulators are designed to model adequately communication protocols, and they can easily scale to large size networks composed of a high number of nodes. Examples of popular network simulators are the open source ns-2, ns-3 and Omnet++, or the commercial solution OPNET modeler. Unfortunately, for clock synchronization, and in particular for those aspects mostly taken into account here (i.e., effects of temperature and in-node latencies on synchronization quality), available network simulators are not adequate. Firstly, they cannot currently model XOs and in-node latencies. Both aspects can be in theory included by modifying simulator models, but this requires a good knowledge of the simulator, a lot of effort, and there is no guarantee that changes will be compatible with newer versions of the simulator. Secondly, network simulators are not designed to model more than one timescale. In synchronization, each node has a different view of the time, and its dependence with respect to the timescale of the simulator is complex (see, e.g., subsections 3.2 and 3.3). Finally, to model the schema proposed in Fig. 1, if each message exchange starts and end in $[k, k+1], \forall k$, a discrete event simulator is not needed. In fact, timestamps acquired by nodes in the exchange started at time $k + 1$ do not depend on that acquired in the exchange k. This reasonable assumption reduces considerably the complexity of the simulator.

For all these reasons, an “ad-hoc” simulation environment has been specifically developed to model the system as described in the previous sections. The simulation software was programmed in `python`, and it was executed in parallel (a process for every value of K) in a High-Performance Computing (HPC) cluster consisting of 544 cores placed in 17 computational nodes, and with a total amount of RAM equal to 2.2 TB. The simulation process is subdivided in two phases. In the first phase, the simulator makes use of state equations (including noises, environmental temperature variation and XO models) described in Section 3, to obtain for each time instant the four timestamps exploited by the clock correction algorithm (i.e, t_1, t_2, t_3 and t_4), and the correct target time. In the second phase, data obtained in the first step are exploited to compare clock correction algorithms. Splitting the simulation into two steps offers a big advantage in terms of execution speed, because data have not to be reproduced each time the performance of a clock correction algorithm has to be tested, and it ensures a fair comparison between algorithms [46], i.e., all are applied to the same data set.



Plot.	In-node latencies	Temperature	Description
a	hw^{WiFi}	$\omega^{norm}_{\gamma^T}(t,\cdot)$	WiFi node with software timestamps and normal temperature variations (between 10 °C and 35 °C every 1200 s)
b	sw^{WSN}	$\omega^{norm}_{\gamma^T}(t,\cdot)$	WSN node with software timestamps and normal temperature variations (between 10 °C and 35 °C every 1200 s)
c	hw^{WSN}	$\omega^{norm}_{\gamma^T}(t,\cdot)$	WSN node with hardware timestamps and normal temperature variations (between 10 °C and 35 °C every 1200 s)
d	hw^{WiFi}	$\omega^{high}_{\gamma^T}(t,\cdot)$	WiFi node with software timestamps and huge temperature variations (between −10 °C and 40 °C every 600 s)
e	sw^{WSN}	$\omega^{high}_{\gamma^T}(t,\cdot)$	WSN node with software timestamps and huge temperature variations (between −10 °C and 40 °C every 600 s)
f	hw^{WSN}	$\omega^{high}_{\gamma^T}(t,\cdot)$	WSN node with hardware timestamps and huge temperature variations (between −10 °C and 40 °C every 600 s)

Figure 4. 99.9-percentile of the synchronization error, for S_1, S_2, S_3, NN and CH ($\omega^{high}_{\gamma^T}(t,\cdot)$ and $\omega^{norm}_{\gamma^T}(t,\cdot)$ temperature models; sw^{WiFi} and sw^{WSN} in-node latencies; $n_\mu = 1$ and $n_\sigma = 1$).

10.2. Varying nodes type and environmental conditions

In the first set of experiments (Fig. 4), we evaluate the performance of all the techniques and temperature models proposed, with the exception of hw^{WiFi}. For the hw^{WSN} condition, the delays $d^{node'}_S$ and d^{node}_R have been distributed uniformly between 0 and 31 μs. This setting makes it possible to put out of phase the sending times of the two exchanged packets. This procedure also removes possible correlations between the timestamps obtained in subsequent synchronization steps.

It is clear from the figure that CH is never optimal. The splines, especially S_1, guarantee the optimal performance only for short ranges of K. The optimal setting (i.e., minimum $p99.9$) of K is denoted with K^*. Under high temperature oscillations (i.e., $\omega^{high}_{\gamma^T}(t,\cdot)$), both the splines and CH have a significant performance degradation as soon as K slightly differs from K^*. The value K^* is not constant and it depends on the spline used, on the temperature model and on the in-node delays. As a consequence, the estimation of K^* is hardly possible. Conversely, the NN guarantees optimal performance for larger ranges of K. Only the most critical condition of Plot 4.e (software timestamping in WSN and high temperature variations) leads to larger performance oscillations in the NN. The method based on NN provides lower synchronization errors also in the case of hardware timestamps with 31 μs resolution (hw^{WSN} condition), outperforming all the other methods.

In Fig. 5, the S_1, S_3 and CH techniques have been analyzed in the proximity of a trend inversion (at time 40 s, the temperature reaches the minimum, $\sim T^{Low}_E$, and starts to increase again). Fig. 5 helps highlight the impact of tracking the variability of the target offset, which is slowly approximated by S_1 and CH. In this case, with $K = 70$, the best tracking is obtained by S_3 and NN (the NN is not reported for the sake of clarity). Under small values of K, however, the use of high order splines is not convenient, because the contribution of the measurement noise

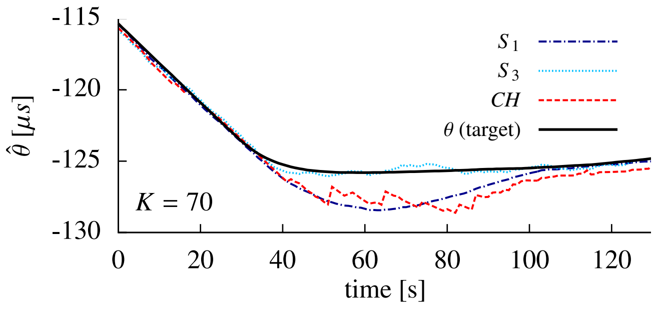


Figure 5. Estimation of the offset ($\hat{\theta}$), with different time-varing techniques, in proximity of a trend inversion at time 40 s ($K = 70$).

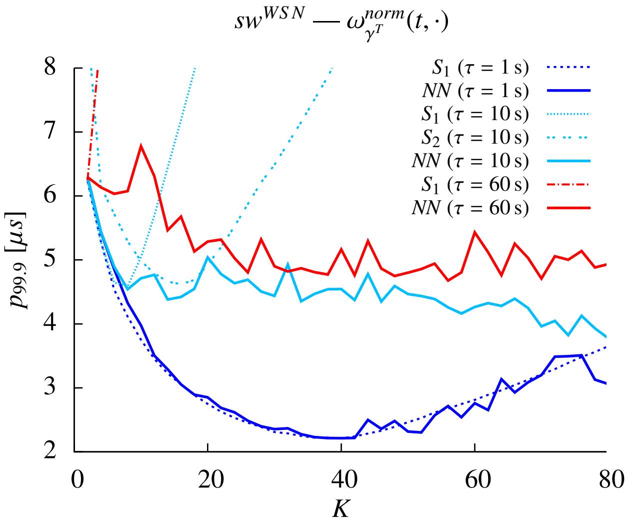


Figure 6. 99.9-percentile of the synchronization error, for S_1 and NN with different values of the synchronization period ($\tau = 1, 10, 60$ s). For $\tau = 10$ s also the results of the S_2 technique has been reported. Experimental condition: sw^{WSN} — $\omega^{norm}_{\gamma^T}(t,\cdot)$, i.e., WSN node with software timestamps and normal temperature variations (between 10 °C and 35 °C every 1200 s).

is predominant with respect to the temperature effect. In virtue of the small sensitivity on K variations, NN is not affected by this problem.

10.3. Synchronization period τ

In many WSN applications, the interval between two adjacent synchronization steps, τ, is usually increased as much as possible to improve battery duration. Unfortunately, an increasing τ (leaving K unchanged) decreases the ability to track temperature variations. To compensate large τ, a reduction of K may be applied, but this would lead to worst in-node delays compensations. The estimation performance thus plays an important role in this perspective as well.

A number of experiments with variable τ have been carried out for the sw^{WSN} scenario, the $\omega^{norm}_{\gamma^T}(t,\cdot)$ temperature model, and with different values of τ, namely $\tau = 1$ s, 10 s, 60 s.

Results reported in Fig. 6 show how synchronization error is directly related to τ. For instance, increasing of one order of magnitude the value τ, from 1 s to 10 s, the error in correspondence to K^* passes from 2.211 μs to 5.590 μs for S_1 and from

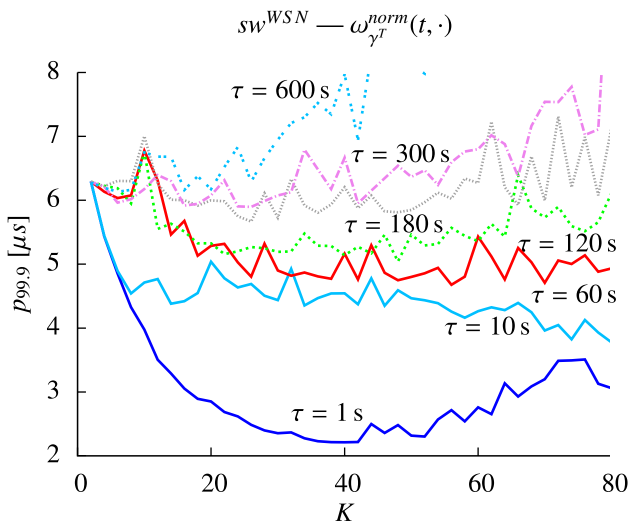


Figure 7. 99.9-percentile of the synchronization error, for NN with different values of the synchronization period ($\tau = 1, 10, 60, 120, 180, 300, 600$ s). Experimental condition: sw^{WSN} — $\omega^{norm}_{\gamma^T}(t,\cdot)$, i.e., WSN node with software timestamps and normal temperature variations (between 10 °C and 35 °C every 1200 s).

2.212 μs to 4.162 μs for NN. After a further increase to $\tau = 60$ s, the synchronization error worsens and it reaches in the point K^* the minimum errors of 6.245 μs and 4.678 μs for S_1 and NN, respectively.

For $\tau = 10$ s, also the statistics related to S_2 has been reported in the plot. The minimum achieved error is 4.621 μs with $K^* = 16$. The errors of S_1 and S_2 are higher than the one of the NN method, for every value of K and τ.

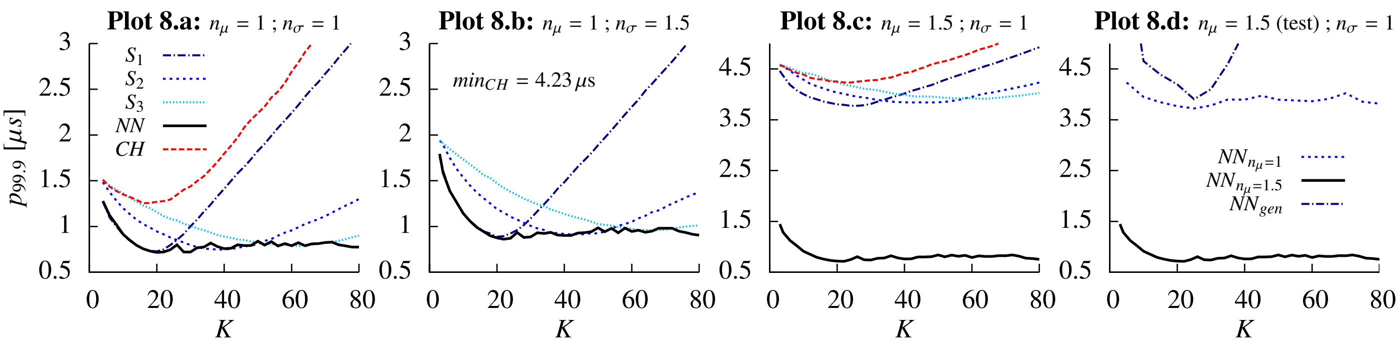
With values of τ greater than a given threshold (i.e., $\tau \geq 180$ s as in Fig. 7), synchronization quality worsen because NN method is no longer able to estimate correctly the temperature variations. In other words, the error achieved by NN in compensating temperature variations is bigger than the error due to in-node delay.

10.4. Delay asymmetry

In Fig. 8, the effect of latency asymmetries on synchronization accuracy are analyzed. Tests have been performed using sw^{WiFi} and $\omega^{norm}_{\gamma^T}(t,\cdot)$ as temperature model. Plot 8.a represents a perfect symmetry where both sender and receiver experience the same latency distributions (i.e., $n_\mu = 1$ and $n_\sigma = 1$ in Table 2). Results for this configuration are equal to those discussed in the previous subsection.

In the second experiment (Plot 8.b), an asymmetry on the width of the gaussian distributions that model the latencies has been introduced by setting $n_\sigma = 1.5$. As expected, the synchronization quality worsen for all the analyzed techniques, because the precision of the timestamps at the receiver has been reduced. The technique mostly affected by the asymmetry is CH.

In the third experiment (Plot 8.c), the asymmetry has been obtained adding a systematic error on timestamps acquisition, by multiplying only the mean value of the gaussian distributions of the receiver node: $n_\mu = 1.5$ and $n_\sigma = 1$. Only the NN



Plot.	n_μ	n_σ	Description
a	1	1	Perfect symmetry between sender and receiver in-node latencies
b	1	1.5	Asymmetry on in-nodes latencies precision was obtained by increasing the width of the receiver node in-node latencies
c	1.5	1	Asymmetry on in-nodes latencies accuracy was obtaining by adding a systematic error in the receiver node in-node latencies
d	1.5 (test)	1	Same test configuration regarding asymmetry of Plot c. Three different train configurations: $NN_{n_\mu=1}$ equal to Plot a; $NN_{n_\mu=1}$ equal to Plot c; NN_{gen} obtained mixing different n_μ values

Figure 8. The effect of in-node asymmetries on the 99.9-percentile of the synchronization error, for S_1, S_2, S_3, NN and CH ($\omega^{norm}_{\gamma^T}(t,\cdot)$ temperature models; sw^{WiFi} in-node latencies).

technique compensates the asymmetry, while other techniques suffer of a degradation on accuracy of about 3.2 μs.

Experimental results reflect directly the systematic accuracy error introduced by the asymmetry. Since the delay sampling of equation (8) does not hold under asymmetry (the multiplication by $\frac{1}{2}$ assumes path symmetry), asymmetry may cause a systematic error, whose correction requires an accurate calibration (as mentioned, for example, in [43]). The calibration may be driven by a-priori calculations as shown in Appendix C. Since a-priori calculations may be hardly applied in practice, the calibration may derive from the direct measurement of the synchronization error. The calibration provided by the *NN*, together with the compensation of other non-stationary effects (e.g., temperature), greatly simplifies the synchronization process.

The fourth experiment (Plot 8.d) analyzes the robustness of *NN* with respect to the asymmetry of the channel. All tests are performed using $n_\mu = 1.5$. Results, reported as $NN_{n_\mu=1}$ and $NN_{n_\mu=1.5}$, denote *NN* models trained with $n_\mu = 1$ and $n_\mu = 1.5$, respectively. For NN_{gen}, the *NN* model has been trained with five data sets composed of 25000 samples that differ on the values of n_μ (0.25, 0.75, 1, 1.25, 1.75, respectively). As expected, *NN* cannot generalize the channel asymmetry as in the case of temperature [15], because formula (8) supposes $d^{path}_{SR} = d^{path}_{RS}$. Basically, when the *NN* is trained by using data sets with different channel asymmetries, it reaches a minimum error only for one of the possible asymmetries. $NN_{n_\mu=1.5}$ has the same systematic error in all the cases. This error can be derived from the calculations presented in Appendix C. When the test and training conditions are consistent from the viewpoint of channel asymmetry, *NN* compensates the error.

10.5. Exponentially distributed delay

For the sake of completeness, we consider also exponentially distributed delays as in [17, 16], which may characterize peculiar scenarios [16]. In the first two experiments of Fig. 9, no in-node delays are considered, while d^{prop}_{SR} and d^{prop}_{RS} are exponentially distributed with mean 1 μs. This condition has been denoted as $exp(\frac{1}{1\mu s})$ in Fig. 9.

In the first Plot 9.a, *CH* achieves the best performance. The *NN* tries to follow the performance of the best spline, which changes from S_1 to S_3 with increasing K.

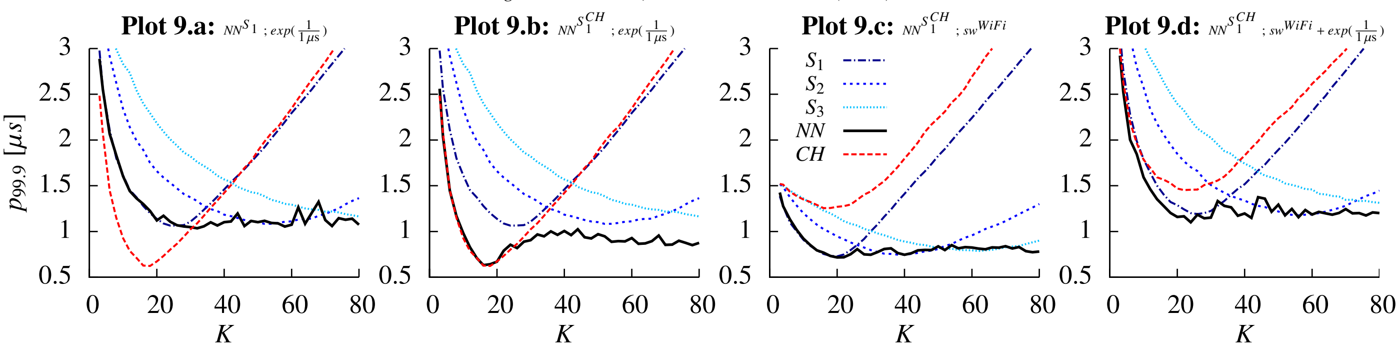
In the second Plot 9.b, the *NN* information vector is collected on the basis of a first-order regression scheme, whose slope and y-intercept coefficients are derived from *CH* (denoted with S^{CH}_1). Results regarding this new information vector have been reported as $NN^{S^{CH}_1}$, while NN^{S_1} identifies those derived from S_1. More specifically, the S^{CH}_1 is obtained from computing the slope and the y-intercept of S_1 through the skew and offset estimated by equations (10) and (11) in [17], respectively. As outlined in Fig. 2, skew and S_1 slope are strictly related. With exponential distributed delay, the $NN^{S^{CH}_1}$ method guarantees the best performance with a larger set of K. The quality of synchronization is always better, regardless the value of K.

The new information vector based on S^{CH}_1 is analyzed in the third experiment (Plot 9.c) with Gaussian in-node delays (sw^{WiFi}) and with $d^{prop}_{SR} = d^{prop}_{RS} = 0$. Surprisingly, the synchronization accuracy of $NN^{S^{CH}_1}$ is comparable with the one of NN^{S_1} (Plot 4.a). This means that the information vector based on S^{CH}_1 generalizes the *NN* technique to both in-node Gaussian and exponential delays.

As a further verification of this property, in the last Plot 9.d, both Gaussian sw^{WiFi} and exponential $exp(\frac{1}{1\mu s})$ delays have been activated in the simulator. Once again, the results highlight the capability of $NN^{S^{CH}_1}$ to generalize to different delays distributions.

10.6. Multi-hop scenario

The sw^{WiFi} and sw^{WSN} conditions have been analyzed as they lead to the lower synchronization quality. Five hops are considered as in [19]. Table 3 lists the parameters of the function $\omega_{\gamma^T}(t,\cdot)$ for the five hops. The parameters assigned to the



Plot.	Information vector	Delay model	Description
a	NN^{S_1}	$exp(\frac{1}{1\mu s})$	Information vector based on S_1, and only exponentially distributed delay
b	$NN^{S_1^{CH}}$	$exp(\frac{1}{1\mu s})$	Information vector based on S_1^{CH}, and only exponentially distributed delay
c	$NN^{S_1^{CH}}$	sw^{WiFi}	Information vector based on S_1^{CH}, and only Gaussian in-node delay
d	$NN^{S_1^{CH}}$	$sw^{WiFi} + exp(\frac{1}{1\mu s})$	Information vector based on S_1^{CH}, and both Gaussian in-node delay and exponentially distributed delay

Figure 9. 99.9-percentile of the synchronization error, with different delays distributions (sw^{WiFi} and $exp(\frac{1}{1\mu s})$), and evaluation of the $NN^{S_1^{CH}}$ method.

Table 3. Environment characterization (i.e., parameters of the function $\omega_{\gamma^T}(t,\cdot) = \omega_{\gamma^T}(t, tc, p, T_E^{High}, T_E^{Low})$) for every hop of the multi-hop scenario.

hop	T_E^{Low} [°C]	T_E^{High} [°C]	tc [s]	p [s]
1	10	35	60	1200
2	15	30	60	1200
3	5	40	60	1200
4	17	33	120	1200
5	10	35	90	900

different hops involve all the parameter of the function $\omega_{\gamma^T}(t,\cdot)$. In the first three hops, changes regard temperature boundaries (i.e., T_E^{Low} and T_E^{High}); in the fourth hop, variations on both temperature boundaries and thermal time constant have been considered; the last hop takes into account a different periodicity (e.g., the node is placed in another operating environment, which is characterized by a different temperature variation period).

Table 4 reports the experimental results. In addition to the $p_{99.9}$, also the standard deviation (σ) and the maximum (Max) of the absolute value of the synchronization error have been provided.

The three methods based on NN, namely NN_{link}, NN_{gen_A} and NN_{gen_B}, have been compared with S_1.

The first column of results of Table 4 regards S_1. For this experiment, the values K^* obtained for the first hop (i.e., 20 for sw^{WiFi} and 40 for sw^{WSN}) have been fixed and used for the remaining four hops. The symbol K^* has been substituted in the result table with the symbol K, because the optimality of its value only applies to the first hop.

All the statistical indexes worsen with the distance from the root. Specifically, in the case of σ, the increase in the hop number is almost constant. The values of these constants are $0.201\,\mu$s for sw^{WiFi} and $0.702\,\mu$s for sw^{WSN}. Roughly speaking, the value of σ doubles every hop. A similar behavior affects $p_{99.9}$ and Max. Their trends with the distance are less deterministic because they are more affected by rare events.

In all the NN methods presented, K must be chosen large enough in order to obtain timestamps jitter compensation; it should be greater than the values K^* obtained for S_1 and for all the possible combinations of timestamps jitters and temperature variations. Fixing $K = 60$ meets the requirement (values of K greater than 60 lead to similar results because the NN is not too sensitive to changes of K).

The NN_{link} method is the best from the point of view of performance because the resulting NNs are specialized to cope with the specific experimental conditions of each couples of nodes, the network channel and the environment. The results reported in the column NN_{link} of Table 4 confirm the ability of the NN to outperform S_1, and the benefit of using NN_{link} increases with the number of hops. Actually, the results regarding the synchronization between N_1 and N_0 in the first hop reflect exactly those reported for the same condition in Fig. 4.

The training database of NN_{gen_A} has been derived by merging the training databases of five separate experiments in which the nodes N_1, N_2, N_3, N_4 and N_5 of Fig. 3 are directly connected with the reference clock N_0. In this case, the training patterns contain only the information regarding temperature variations and the in-node delays, but not the hop error. The resulting database contains 1500000 patterns. As expected, all the performance indicators have worse performance than in the NN_{link} case, but, excluding the maximum value of the sw^{WiFi} condition for the hop number 5, the synchronization quality of NN_{gen_A} is similar or outperforms the one of S_1.

Under NN_{gen_B}, the temperature conditions 5, 4, 3, 2, 1 of Table 3 have been used, in the reported inverted order from 5 to 1, to train the 5 hops of the network presented in Fig. 3,

Table 4. Synchronization quality in a multi-hop scenario for S_1, NN_{link} (i.e, NN trained for every hop), and NN_{gen_A} and NN_{gen_B} (i.e., the same NN for every hop).

	Hop		S_1				NN_{link}			NN_{gen_A}			NN_{gen_B}		
		K	σ	$p_{99.9}$	Max	K	σ	$p_{99.9}$	Max	σ	$p_{99.9}$	Max	σ	$p_{99.9}$	Max
WiFi (sw^{WiFi})	1	20	0.207	0.717	1.184	60	0.186	0.761	1.202	0.183	0.772	1.192	0.198	0.832	1.262
	2	20	0.400	1.339	2.103	60	0.280	1.090	1.669	0.330	1.245	1.805	0.308	1.136	1.988
	3	20	0.635	2.369	3.946	60	0.432	1.553	2.413	0.528	2.180	3.045	0.477	1.862	3.176
	4	20	0.855	3.060	5.403	60	0.493	1.673	2.359	0.711	2.802	5.030	0.561	2.093	4.039
	5	20	1.110	3.846	6.415	60	0.600	2.006	2.734	0.946	3.854	11.052	0.668	2.567	4.972
WSN (sw^{WSN})	1	40	0.633	2.205	3.349	60	0.647	2.794	3.479	0.623	2.267	3.104	0.676	2.328	3.018
	2	40	1.200	4.001	5.658	60	1.112	4.270	5.754	1.080	3.926	5.254	1.162	4.049	5.110
	3	40	1.947	7.111	10.200	60	1.561	5.290	7.314	1.860	7.089	9.176	1.720	6.020	7.786
	4	40	2.630	9.347	13.158	60	1.967	6.478	8.460	2.390	9.009	11.016	2.105	7.358	9.190
	5	40	3.440	11.831	16.825	60	2.355	7.626	10.500	2.999	11.198	13.862	2.484	8.744	11.163

All values are expressed in μs.

respectively. The resulting database contains 1500000 patterns.

The results reported in the last column of Table 4 show that NN_{gen_B} performance is very closed to that of NN_{link}. This tells us two things: that the use of a generalized *NN* is possible because results are close enough to the optimum represented by NN_{link}, and that the knowledge of the hop error is important because it drives to a better performance.

10.7. Testbed

Implementation and validation of synchronization protocols in real devices deserve specific attention. Results reproducibility is the first issue to be considered. Some experimental conditions may be difficult to be reproduced (i.e., temperature variations, in-node latencies or packet losses) as well as experiments duration that may be limited. The interested reader is referred to [47] for details on reproducibility issues. How to measure the synchronization quality is another concern. In [48], the generation of two synchronous signals is used, together with an external device to check their time differences. Unfortunately, real devices add variable jitters when they trigger an actuation. In order to minimize this jitter, a hard real-time implementation is required. Moreover, a temperature-controlled environment may be also used to analyze temperature variations (see, e.g., [21]).

Some preliminary results of the *NN* on the real implementation of [48] are summarized here. Besides the fact that temperature variations are lower than the ones considered in the simulations (due to the heating system of the room considered), we obtained interesting results that confirm the applicability of the approach. The RBIS protocol [6] is used to acquire two databases with respect to working days [47] and weekend, respectively. The performance of the *NN* is qualitatively comparable with the one obtained in simulations here when compared to the splines. An interesting result is that the *NN* outperforms S_1 for small value of K. This is useful in a WSN device because it reduces the size of the information vector and the inherent computation. Another promising issue relies on the fact that the *NN* shows to be robust to loss of packets. This may open the door to outperforming other estimation approaches, specifically designed for intermittent observations, still being based on the Gaussian hypothesis. On the other hand, the *NN* fails when trained on the week days and tested over the weekend. In order to prevent such a performance degradation, one may anticipate the working conditions of the *NN*, in particular with respect to the temperature ranges to be addressed. This was validated by results not reported here for the sake of conciseness.

11. Conclusions and Future Work

We have examined and discussed a neural estimation technique for the popular two-way timing message exchange synchronization protocol and for nodes affected by temperature variations and delay asymmetry. The impacts of the delay knowledge and the presence of several hops in the network have been accurately analyzed. Numerical analysis reveals significant performance improvements over existing techniques (splines and [17]) under variable temperatures and different delay distributions. One of the most important outcomes is the robustness to increasing synchronization steps (high accuracy, independently to the number of timestamps used).

Future work includes different topics. The skew estimation and the robustness to loss of timestamps are currently under investigation. Other intriguing issues are: the impact of the knowledge of the temperature through observations, runtime retraining of the neural estimator to unexpected conditions, as well as the management of the multi-hop database under "big data" paradigms. Preliminary results on a real implementation [48] confirm the effectiveness of the proposed technique. A more in-depth validation on a number of real installations and environmental conditions is argument of future research as well.

Acknowledgment

Computational resources were provided by HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it).

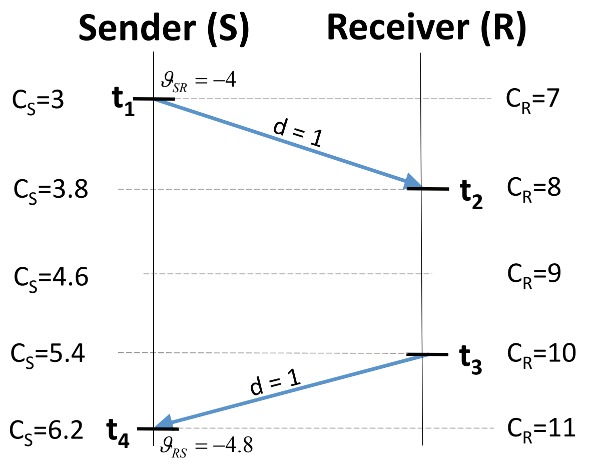


Figure A.10. Example of application of the measurement equations.

Appendix A.

An example for explaining the measurement equations (8)-(10) of subsection 3.4 has been reported in Fig. A.10. The index k has been removed for the sake of simplicity and because only one message exchange is analyzed. The quantity C_R is the free-running clock register of the *receiver* node, which holds the reference time. Instead, C_S (i.e., the free-running clock register of the *sender* node) is updated with a lower frequency than C_R. The timestamps exploited for synchronization are: $t_1 = 3$, $t_2 = 8$, $t_3 = 10$, $t_4 = 6.2$. The value γ has been initialized with the estimation of the skew performed in the previous $k-1$ synchronization step (i.e., $\hat{\gamma} = -0.25$). The new estimation of the delay from (8) is:

$$\begin{aligned}\hat{d} &= \frac{(1-\hat{\gamma})\cdot(t_4-t_1)-(t_3-t_2)}{2} = \\ &= \frac{1.25\cdot(6.2-3)-(10-8)}{2} = \frac{4-2}{2} = 1\end{aligned} \tag{A.1}$$

By using the value $\hat{d}$ computed in (A.1), the new estimations of the offsets and of the skew can be obtained from (9) and (10):

$$\begin{aligned}\hat{\theta}_{SR} &= t_1-(t_2-d) = 3-8+1 = -4 \quad (A.2)\\ \hat{\theta}_{RS} &= t_4-(t_3+d) = 6.2-10-1 = -4.8\\ \hat{\gamma} &= 1-\left(\frac{t_3+d-(t_2-d)}{t_4-t_1}\right) = 1-\left(\frac{11-7}{6.2-3}\right) = (A.3)\\ &= 1-\frac{4}{3.2} = 1-1.25 = -0.25\end{aligned}$$

Appendix B.

We demonstrate here that 1-st order regression applied to timestamps is independent to the delay correction outlined in subsection 5.1.

The parameters of the 1-st order regression (S_1) interpolating a set of two-dimensional points $(X_k, Y_k), k = 1, ..., K$, i.e., slope m and y-intercept q, are:

$$m = \frac{\sum_{k=1}^K X_k Y_k - \frac{1}{K}\sum_{k=1}^K X_k \sum_{k=1}^K Y_k}{\sum_{k=1}^K X_k^2 - \frac{1}{K}(\sum_{k=1}^K X_k)^2}; q = \bar{Y} - m\bar{X} \tag{B.1}$$

Two straight lines are compared, with and without delay correction (d). The timestamps without correction consist of the sequence $\{t_1^k, t_2^k, t_3^k, t_4^k\}, k = 1, ..., K$; let m and q be the resulting S_1 parameters. The timestamps with correction are $\{t_1^k, t_2^{k'}, t_3^{k'}, t_4^k\}$, $k = 1, ..., K$, $t_2^{k'} = t_2^k - d$, $t_3^{k'} = t_3^k + d$ with m' and q' the corresponding S_1' parameters. The two sets of parameters asymptotically converge (in the number of timestamps) to identical values. If $m \sim m'$ ('$\sim$' means asymptotical equivalence), $q \sim q'$ because the averages $\bar{Y}$ and $\bar{X}$ (in B.1) are identical in the two cases. In order to show $m \sim m'$, we substitute in (B.1) the timestamps and consider the difference $\Delta_m = m' - m$ as $K \to +\infty$. After some algebraic manipulations, we obtain:

$$\Delta_m = \frac{d\sum_{k=1}^K (t_4^k - t_1^k)}{\sum_{k=1}^K ((t_1^k)^2 + (t_4^k)^2) - \frac{1}{K}(\sum_{k=1}^K (t_1^k + t_4^k))^2} \tag{B.2}$$

The Δ_m quantity tends to 0 for the following reasons. The numerator is positive ($t_4^k > t_1^k, \forall k$) and grows linearly. The denominator defines a definite positive quadratic form of the timestamps in $\Re^{+K}$; the determinant of the corresponding Hessian matrix is $(2^k \cdot \frac{(k-1)^k - 1}{k^k}) > 0, \forall k > 2$.

This asymptotic behavior is measured in practice with a small number of timestamps. For example, in the sw^{WiFi} condition with $\tau = 1$ s and $K = 60$, $\Delta_m = 1.8 \cdot 10^{-9}$ and $\Delta_q = q' - q = -54$ ns[7]. The difference of the synchronization error between S_1' and S_1 is 54 ns. By doubling the number of timestamps (i.e., $K = 120$) the difference is halved, i.e., it amounts to 27 ns, with $\Delta_m = 4.5 \cdot 10^{-10}$ (an order of magnitude lower than with $K = 60$) and $\Delta_q = -27$ ns.

Appendix C.

The asymmetry error outlined in subsection 10.4 can be a-priori computed by considering the propagation delays between the sender and the receiver nodes ($\overline{d}_{SR}^{path}$), and the one in the opposite direction ($\overline{d}_{RS}^{path}$):

$$\begin{aligned}\overline{d}_{SR}^{path} &= \overline{d}_{SR}^{prop} + \overline{d}_R^{rec} \cdot n_\mu - \overline{d}_S^{send} = \\ &= 0 + 7.23\,\mu\text{s} \cdot 1.5 - 5.4\,\mu\text{s} = 5.45\,\mu\text{s}\\ \overline{d}_{RS}^{path} &= \overline{d}_{RS}^{prop} + \overline{d}_S^{rec} - \overline{d}_R^{send} \cdot n_\mu = \\ &= 0 + 7.23\,\mu\text{s} - 5.4\,\mu\text{s} = -0.87\,\mu\text{s}\end{aligned}$$

As a consequence, the systematic error on the estimation of the propagation delay is:

$$\overline{\epsilon}^{\check{d}(k)} = \frac{\overline{d}_{SR}^{path} - \overline{d}_{RS}^{path}}{2} = \frac{5.45\,\mu\text{s} - 0.87\,\mu\text{s}}{2} = 3.16\,\mu\text{s} \tag{C.1}$$

[7]The value $d = d^{prop} + d^{rec} - d^{send} = 2.16\,\mu$s has been obtained with $d^{rec} = \mu^{rec} = 7.23\,\mu$s, $d^{send} = \mu^{send} = 5.4\,\mu$s and $d^{prop} = 334$ ns. d^{prop} is the time the light takes to cover a distance of 100 m. The quantity d_R^{node} has been set to 0.5 s. Values larger than 0.5 s lead to higher differences (Δ_m, Δ_q) and worse synchronization accuracy, the opposite holds for values smaller than 0.5 s.

References

[1] G. Cena, I. Cibrario Bertolotti, S. Scanzio, A. Valenzano, C. Zunino, Synchronize your watches: Part I: General-purpose solutions for distributed real-time control, IEEE Industrial Electronics Magazine 7 (1) (2013) 18–29. doi:10.1109/MIE.2012.2232354.

[2] A. R. Swain, R. Hansdah, A model for the classification and survey of clock synchronization protocols in WSNs, Ad Hoc Networks 27 (0) (2015) 219–241. doi:http://dx.doi.org/10.1016/j.adhoc.2014.11.021.

[3] Y. Wang, J. Huang, L. Yang, Y. Xue, TOA-based joint synchronization and source localization with random errors in sensor positions and sensor clock biases, Ad Hoc Networks 27 (0) (2015) 99–111. doi:http://dx.doi.org/10.1016/j.adhoc.2014.12.001.

[4] C. Medina, J. Segura, A. de la Torre, Accurate time synchronization of ultrasonic TOF measurements in IEEE 802.15.4 based wireless sensor networks, Ad Hoc Networks 11 (1) (2013) 442–452. doi:http://dx.doi.org/10.1016/j.adhoc.2012.07.005.

[5] M. Johas Teener, A. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, K. Stanton, Heterogeneous networks for audio and video: Using ieee 802.1 audio video bridging, Proceedings of the IEEE 101 (11) (2013) 2339–2354.

[6] G. Cena, S. Scanzio, A. Valenzano, C. Zunino, Implementation and Evaluation of the Reference Broadcast Infrastructure Synchronization Protocol, IEEE Transactions on Industrial Informatics 11 (3) (2015) 801–811. doi:10.1109/TII.2015.2396003.

[7] K.-H. Phung, B. Lemmens, M. Goossens, A. Nowe, L. Tran, K. Steenhaut, Schedule-based multi-channel communication in wireless sensor networks: A complete design and performance evaluation, Ad Hoc Networks 26 (0) (2015) 88–102. doi:http://dx.doi.org/10.1016/j.adhoc.2014.11.008.

[8] A. Berger, M. Pichler, J. Klinglmayr, A. Potsch, A. Springer, Low-Complex Synchronization Algorithms for Embedded Wireless Sensor Networks, IEEE Transactions on Instrumentation and Measurement 64 (4) (2015) 1032–1042. doi:10.1109/TIM.2014.2366272.

[9] D. Djenouri, N. Merabtine, F. Z. Mekahlia, M. Doudou, Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks, Ad Hoc Networks 11 (8) (2013) 2329–2344. doi:http://dx.doi.org/10.1016/j.adhoc.2013.06.001.

[10] Y.-C. Wu, Q. Chaudhari, E. Serpedin, Clock Synchronization of Wireless Sensor Networks, IEEE Signal Processing Magazine 28 (1) (2011) 124–138. doi:10.1109/MSP.2010.938757.

[11] G. Giorgi, C. Narduzzi, Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks, IEEE Transactions on Instrumentation and Measurement 60 (8) (2011) 2902–2909. doi:10.1109/TIM.2011.2113120.

[12] J. R. Vig, Quartz Crystal Resonators and Oscillators; For Frequency Control and Timing Applications - A Tutorial, Tutorial, US Army Communications-Electronics Research, Development & Engineering Center Fort Monmouth, NJ, USA, Rev. 8.5.2.2, 2004.

[13] Z. Yang, L. He, L. Cai, J. Pan, Temperature-Assisted Clock Synchronization and Self-Calibration for Sensor Networks, IEEE Transactions on Wireless Communications 13 (6) (2014) 3419–3429. doi:10.1109/TWC.2014.051414.130270.

[14] Z. Yang, L. Cai, Y. Liu, J. Pan, Environment-aware clock skew estimation and synchronization for wireless sensor networks, in: Proceedings IEEE INFOCOM, 2012, pp. 1017–1025. doi:10.1109/INFCOM.2012.6195457.

[15] M. Mongelli, S. Scanzio, Approximating Optimal Estimation of Time Offset Synchronization With Temperature Variations, IEEE Transactions on Instrumentation and Measurement 63 (12) (2014) 2872–2881. doi:10.1109/TIM.2014.2320400.

[16] W. Sun, F. Brannstrom, E. Strom, On clock offset and skew estimation with exponentially distributed delays, in: IEEE International Conference on Communications (ICC 2013), 2013, pp. 1872–1877. doi:10.1109/ICC.2013.6654794.

[17] Q. Chaudhari, A Simple and Robust Clock Synchronization Scheme, IEEE Transactions on Communications 60 (2) (2012) 328–332. doi:10.1109/TCOMM.2011.110711.100136.

[18] G. Giorgi, An Event-Based Kalman Filter for Clock Synchronization, IEEE Transactions on Instrumentation and Measurement 64 (2) (2015) 449–457. doi:10.1109/TIM.2014.2340631.

[19] J. J. Pérez-Solano, S. Felici-Castell, Adaptive time window linear regression algorithm for accurate time synchronization in wireless sensor networks, Ad Hoc Networks 24, Part A (0) (2015) 92–108. doi:http://dx.doi.org/10.1016/j.adhoc.2014.08.002.

[20] H. Zhou, T. Kunz, H. Schwartz, Adaptive correction method for an OCXO and investigation of analytical cumulative time error upper bound, IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 58 (1) (2011) 43–50. doi:10.1109/TUFFC.2011.1772.

[21] M. Xu, W. Xu, T. Han, Z. Lin, Energy-efficient time synchronization in wireless sensor networks via temperature-aware compensation, ACM Trans. Sen. Netw. 12 (2) (2016) 12:1–12:29. doi:10.1145/2876508. URL http://doi.acm.org/10.1145/2876508

[22] T. Yang, Y. Dong, X. Zhang, Frequency tracking synchronization algorithm for high latency wireless sensor networks, in: 47th Annual Conference on Information Sciences and Systems (CISS), 2013, pp. 1–5. doi:10.1109/CISS.2013.6624255.

[23] S. Lv, Y. Lu, Y. Ji, An Enhanced IEEE 1588 Time Synchronization for Asymmetric Communication Link in Packet Transport Network, IEEE Communications Letters 14 (8) (2010) 764–766. doi:10.1109/LCOMM.2010.08.091601.

[24] S. Lee, An Enhanced IEEE 1588 Time Synchronization Algorithm for Asymmetric Communication Link using Block Burst Transmission, IEEE Communications Letters 12 (9) (2008) 687–689. doi:10.1109/LCOMM.2008.080824.

[25] S. Lee, S. Lee, C. Hong, An Accuracy Enhanced IEEE 1588 Synchronization Protocol for Dynamically Changing and Asymmetric Wireless Links, IEEE Communications Letters 16 (2) (2012) 190–192. doi:10.1109/LCOMM.2011.092011.110582.

[26] R. Exel, Mitigation of Asymmetric Link Delays in IEEE 1588 Clock Synchronization Systems, IEEE Communications Letters 18 (3) (2014) 507–510. doi:10.1109/LCOMM.2014.012214.132540.

[27] Z. Chaloupka, N. Alsindi, J. Aweya, Clock Synchronization Over Communication Paths With Queue-Induced Delay Asymmetries, IEEE Communications Letters 18 (9) (2014) 1551–1554. doi:10.1109/LCOMM.2014.2341604.

[28] M. Hajikhani, T. Kunz, H. Schwartz, A recursive method for bias estimation in asymmetric packet-based networks, in: Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2014 IEEE International Symposium on, 2014, pp. 47–52. doi:10.1109/ISPCS.2014.6948530.

[29] A. S. Daniel R. Jeske, Estimation of clock offset using bootstrap bias-correction techniques, Technometrics 45 (3) (2003) 256–261. URL http://www.jstor.org/stable/25047052

[30] A. C. Daniel R. Jeske, Effectiveness of bootstrap bias correction in the context of clock offset estimators, Technometrics 48 (4) (2006) 530–538. URL http://www.jstor.org/stable/25471245

[31] G. Huang, A. Y. Zomaya, F. C. Delicato, P. F. Pires, Long term and large scale time synchronization in wireless sensor networks, Computer Communications 37 (2014) 77 – 91. doi:http://dx.doi.org/10.1016/j.comcom.2013.10.003.

[32] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync Protocol for Sensor Networks, in: 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003), ACM, 2003, pp. 138–149.

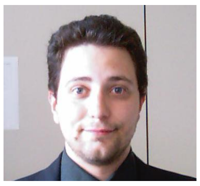
[33] IEEE, IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007) (2012) 1–2793doi:10.1109/IEEESTD.2012.6178212.

[34] M. Mock, R. Frings, E. Nett, S. Trikaliotis, Continuous clock synchronization in wireless real-time applications, in: 19th IEEE Symposium on Reliable Distributed Systems (SRDS), 2000, pp. 125–132. doi:10.1109/RELDI.2000.885400.

[35] R. Bechmann, A. Ballato, T. Lukaszek, Higher-Order Temperature Coefficients of the Elastic Stiffinesses and Compliances of Alpha-Quartz, Proceedings of the IRE 50 (8) (1962) 1812–1822. doi:10.1109/JRPROC.1962.288222.

[36] M. A. T. Figueiredo, Lecture Notes on Bayesian Estimation and Classification, ETH, Accessed: 2016-02-04 (2004). URL http://www.lx.it.pt/$\sim$mtf/learning/Bayes_lecture_notes.pdf

[37] W. H. Greene, Econometric analysis, Prentice Hall, 2002.
[38] H. Flatt, S. Schriegel, J. Jasperneite, Reliable synchronization accuracy in IEEE 1588 networks using device qualification with standard test patterns, in: IEEE International Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2013, pp. 107–112. doi:10.1109/ISPCS.2013.6644772.
[39] D. C. Montgomery, E. A. Peck, G. G. Vining, Introduction to Linear Regression Analysis, 5th Edition, Wiley, 2012.
[40] S. Scanzio, S. Cumani, R. Gemello, F. Mana, P. Laface, Parallel Implementation of Artificial Neural Network Training for Speech Recognition, Pattern Recogn. Lett. 31 (11) (2010) 1302–1309. doi:10.1016/j.patrec.2010.02.003.
[41] ISO/IEC, International Vocabulary of Metrology - Basic and General Concepts and Associated Terms (VIM), ISO/IEC Guide 99-12:2007 (2007) 1–92.
[42] C. A. Boano, N. Tsiftes, T. Voigt, J. Brown, U. Roedig, The impact of temperature on outdoor industrial sensornet applications, IEEE Transactions on Industrial Informatics 6 (3) (2010) 451–459. doi:10.1109/TII.2009.2035111.
[43] A. Mahmood, R. Exel, T. Sauter, Delay and Jitter Characterization for Software-Based Clock Synchronization Over WLAN Using PTP, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1198–1206. doi:10.1109/TII.2014.2304413.
[44] P. A. Sommer, Wireless embedded systems. time, location, and applications, ETH, Accessed: 2014-09-25 (2011).
URL http://dx.doi.org/10.3929/ethz-a-006712401
[45] M. Maróti, B. Kusy, G. Simon, A. Lédeczi, The Flooding Time Synchronization Protocol, in: 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), ACM, 2004, pp. 39–49.
[46] G. Cena, S. Scanzio, A. Valenzano, Reliable comparison of clock discipline algorithms for time synchronization protocols, in: IEEE Conference on Emerging Technologies Factory Automation (ETFA), 2015, pp. 1–9. doi:10.1109/ETFA.2015.7301461.
[47] G. Cena, S. Scanzio, A. Valenzano, Reliable comparison of clock discipline algorithms for time synchronization protocols, in: 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA), 2015, pp. 1–9. doi:10.1109/ETFA.2015.7301461.
[48] G. Cena, S. Scanzio, A. Valenzano, C. Zunino, Implementation and evaluation of the reference broadcast infrastructure synchronization protocol, IEEE Transactions on Industrial Informatics 11 (3) (2015) 801–811. doi:10.1109/TII.2015.2396003.



Maurizio Mongelli got his Ph.D. at the University of Genoa in 2004. During the PhD and in the subsequent years, he worked on Quality of Service for military networks and Ethernet resilience for Selex Communications. He spent three months working on a project on satellite emulator systems at the German Aerospace Centre in Munich, Germany. He is co-author of over 70 scientific works, including international journals, conferences and patents. His main research activity concerns control of networks, machine learning and cybersecurity.

Stefano Scanzio received the Laurea and Ph.D. degrees in computer science from Politecnico di Torino, Italy, in 2004 and 2008, respectively. From 2004 to 2009, he was involved with the Politecnico di Torino in research on speech recognition and classification methods and algorithms.

Since 2009, he was with the National Research Council of Italy (CNR), where he is a Tenured Technical Researcher with the IEIIT institute. He teaches several courses on computer science. He has authored/co-authored several papers in international journals and conferences in the area of synchronization protocols, industrial communication systems and real-time networks.