A Servo-Clock Model for Chains of Transparent Clocks Affected by Synchronization Period Jitter

Daniele Fontanelli, *Member, IEEE*, David Macii, *Member, IEEE*, Stefano Rinaldi, *Member, IEEE*, Paolo Ferrari, *Member, IEEE*, and Alessandra Flammini, *Member, IEEE*

***Abstract*— Industrial networks for distributed monitoring, control, and automation purposes require high-accuracy clock synchronization in topologies including long chains of cascaded nodes. Unfortunately, accuracy typically degrades as the number of devices and the distance from the synchronization reference node (i.e., the *master* or *grandmaster*) grows, because of the accumulation of multiple uncertainty contributions. To mitigate this problem, the so-called transparent clocks are used in some synchronization protocols, such as the precision transparent clock protocol used in PROFINET IO isochronous real time networks and the precision time protocol version 2, standardized as IEEE 1588-2008. In this paper, an optimal servo-clock in the mean square sense is proposed. The controller relies on both a Kalman filter that estimates the clock state difference with respect to the master and a static-state feedback assuring mean square stability even under the effect of significant fluctuations of the synchronization period. Several multiparametric simulation results in a case study based on the features of PROFINET IO devices confirm that excellent performance can be achieved with the proposed approach.**

***Index Terms*— Industrial control, Kalman filters (KFs), optimal control, synchronization, time measurement, uncertainty.**

I. Introduction

Clock synchronization of networked devices is useful in several applications, such as industrial automation [1], [2], finance [3], and smart grid monitoring and control [4]. Clock synchronization can be implemented either with hardware-only [5], or purely software resources [6]. While best performances are typically achieved in Ethernet-based wired networks, the case of wireless solutions has gained an increasing interest in the last years [7], [8]. The precision time protocol (PTP) (along with its profiles) is at the moment one of the most diffused and accepted solutions to achieve clock synchronization in distributed systems [9]. In various industrial contexts, real-time Ethernet (RTE) communication protocols rely on clock synchronization to make distributed control systems and automation plants faster, more robust and more precise. RTE solutions coincide with standard Ethernet at the physical layer, but they include special MAC layers able both to schedule packet transmission and to manage data traffic according to different priority requirements at the application level.

As known, synchronization uncertainty in long chains of nodes unavoidably grows due to the accumulation of multiple contributions, such as phase and frequency noises, imperfect propagation and communication delay compensation, variable traffic conditions, and packet losses. To mitigate this problem, the state of each clock as well as line and bridge delays have to be estimated with high accuracy [2]. A classic clock state estimator for synchronization purposes is the Kalman filter (KF). Its standalone behavior, along with its advantages and disadvantages, has been thoroughly analyzed as a function of different parameters, both with and without a servo-clock [10]. Scalability and performance issues in the case of multiple cascaded nodes are explicitly addressed in [11], where a space-state model and a different KF for PTP synchronization is introduced. However, not all uncertainty contributions are included in that model. Further studies on the same topic, are instead reported in [12] and [13], where an estimator based on a three-state model (including the local time, the clock rate, and the oscillator frequency drift) has been proposed and validated. Using this model, the state of each clock can be estimated even under the influence of time-varying and harsh environmental conditions.

This paper considerably extends the theoretical study and the results presented in [14], which indeed is mainly focused on the case of PROFINET IO networks. In this paper, the oscillator frequency drift is neglected, since it is not very relevant at room temperature and over reasonably short time intervals [15]. Moreover, an additional optimal controller is used to discipline every clock of the chain. This idea is not completely new, since it was already adopted in [16], where a KF is combined with an optimal linear quadratic regulator (LQR) to achieve better synchronization performance. However, in that model the synchronization period jitter due to switch behavior and network traffic was not considered. From a control perspective, the time variability of measured quantities dramatically affects the quality of control (QoC) and sometimes it may lead to instability [17]. To tackle this problem, different delay-resilient network solutions [18], [19], event-based approaches [20]–[22], or anytime controllers [23], [24] have been proposed. Stemming from these results, a novel

Manuscript received June 14, 2013; revised August 28, 2013; accepted October 1, 2013. Date of publication January 10, 2014; date of current version April 3, 2014. This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under Grant FP7-ICT-257462 HYCON2 NoE. The Associate Editor coordinating the review process was Dr. Deniz Gurkan.

D. Fontanelli and D. Macii are with the Department of Industrial Engineering, University of Trento, Trento 38100, Italy (e-mail: daniele.fontanelli@unitn.it; david.macii@unitn.it).

S. Rinaldi, P. Ferrari, and A. Flammini are with the Department of Information Engineering, University of Brescia, Brescia 25121, Italy (e-mail: stefano.rinaldi@ing.unibs.it; paolo.ferrari@ing.unibs.it; alessandra.flammini@ing.unibs.it).

Digital Object Identifier 10.1109/TIM.2013.2286871

controller that explicitly takes into account synchronization period variability and guarantees a certified level of QoC is defined in this paper. In the following, at first, in Section III, a quite general model describing clock behavior as well as communication delays is introduced. Then, in Section IV the clock state, line delay, and bridge delay estimators are defined and justified. Section V deals with the controller design criteria. Finally, in Section VI some meaningful simulation results based on the features of PROFINET IO networks and extracted from some real hardware devices are reported. Such results are compared also with the performances of more typical servo clocks based on a proportional-integral (PI) controller, under different jitter conditions.

II. Synchronization Protocols Overview

In all Ethernet-based networks where clock synchronization is based on either PTP or precision transparent clock protocol (PTCP), periodic messages or frames containing a reference timestamp are forwarded along the branches of a tree (rooted in the *master* or *grandmaster* and extracted from the original topology) till reaching the leaves of the network. Often, just the so-called *Sync* frames are propagated throughout the network with a nominal period T_s (*one-step* clock model). However, to reduce timestamping uncertainty on the transmission side, the so-called *two-step* model can be adopted. Using this approach, the local time value when a *Sync* frame is physically put on the wire is recorded and it is encapsulated into a subsequent *Follow_up* message, which is sent as soon as possible, i.e., immediately after the corresponding *Sync* (namely with a nominal period T_s as well). Thus, clock synchronization relies on the timestamp included in the *Follow_up* frame and the value in the *Sync* is discarded. This approach generally improves synchronization accuracy at the expense of a larger network traffic. Of course, any *Sync/Follow_up* message experiences a latency that depends on: 1) the type and number of switches crossed my the message; 2) the amount of data traffic handled by every switch; and 3) the line delays between pairs of switches.

In the presence of cascaded nodes, two different general policies can be used to synchronize the leaves of the tree to the master or grandmaster. The first approach is used in both PTPv1 and PTPv2 when the so-called *boundary clocks* (BC) are employed [9], and it relies on the progressive synchronization of the chain of clocks along the branches of the tree. In this case, the timestamps appended to *Sync* or *Follow_up* frames are no longer the grandmaster values, since they are read from the local clock *after* it is disciplined so as to be synchronized to the clock of the father node. Also, the delay between each pair of cascaded nodes is estimated and compensated by measuring half of the round trip time (RTT) associated with the exchange of any pair of frames *Delay_Req/Delay_Res*. This exchange occurs with a period T_{dq}. Generally, $T_{dq} \leq T_s$, because the network topology is assumed to be fixed or just slowly changing. Hence, frequent *line delay* measurements are unnecessary.

The second synchronization policy relies on the so-called *transparent clocks* (TCs). Such nodes are not necessarily synchronized to the master, but they are able to measure the ingress–egress message latency spent in crossing the node (namely the *bridge delay*). Also, they append the total delay accumulated from previous TCs in a specific field of the frame to be forwarded, possibly compensating the frequency offset between clocks.

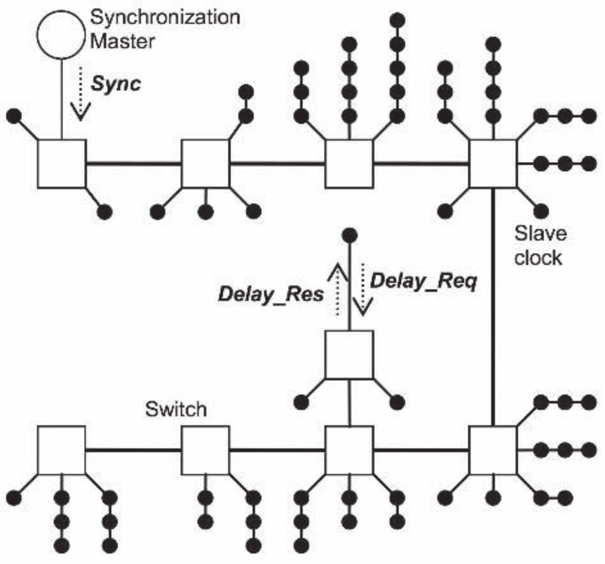


Fig. 1. Starred tree topology of an industrial network.

The total *line delay* between the master and a leaf node can be estimated periodically in two ways, i.e., by exchanging separate *Delay_Req/Delay_Res* frames between pairs of consecutive nodes so that every node can compensate the *line delay* just from the previous node (*peer-to-peer* approach) or by forwarding every pair of *Delay_Req/Delay_Res* frames from a leaf node to the master and backwards through the whole chain, while compensating the intermediate *bridge delays*, as it is done for *Sync/Follow_up* frames (*end-to-end* approach). The former strategy is used both in PTCP and PTPv2 and is generally more flexible and scalable. The latter is described just in PTPv2, but it is not very used in practice, as it may create potential traffic bottlenecks around the synchronization master collecting the *Delay_Res* frames from all leaf nodes. Also, in networks with long linear paths, the solution based on peer-to-peer TCs is preferable to the use of BCs, since cascaded servo-clocks may introduce persistent and large fluctuations, which degrade convergence time and synchronization accuracy [25]. Therefore, in the rest of this paper, just the case of chains of peer-to-peer TCs will be considered. Moreover, for the sake of simplicity, only one-step clocks will be used (i.e., without using *Follow_up* frames). However, the proposed approach can be easily extended to the case of two-step clocks.

III. Model Description

Many industrial networks for automation purposes (e.g., those based on PROFINET IO) exhibit a starred tree topology, as shown in Fig. 1. If we assume that each network node is provided with its own clock and that flicker phase and frequency noises have a negligible effect over short time intervals [10], the behavior of the ith clock (with $i = 0, \dots, N$) can be described by the following linear system

$$\begin{aligned}\dot{\mathbf{x}}_i(t) &= A\mathbf{x}_i(t) + B\mathbf{s}_i(t) + \mathbf{n}_i(t) + \mathbf{u}_i(t)\\ c_i(t) &= C\mathbf{x}_i(t) + \epsilon_i(t)\end{aligned} \tag{1}$$

where

$$A = \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}, \quad B = \begin{bmatrix}1\\ 0\end{bmatrix}, \quad C = [0\ 1].$$

1) t denotes the ideal time on an ideally perfect timescale.
2) $\mathbf{x}_i(t) = [\rho_i(t), \tau_i(t)]^T$ is the system state vector composed by the normalized *clock rate* $\rho_i(t)$ (i.e., ideally equal to one) and by the *time* $\tau_i(t)$ measured by the clock.
3) $\mathbf{n}_i(t) = [n_{\rho_i}(t), n_{\tau_i}(t)]^T$ includes two stationary Gaussian noise sources $n_{\rho_i}(t) \sim \mathcal{N}(\mu_{\rho_i}, \sigma_{\rho_i})$ and $n_{\tau_i}(t) \sim \mathcal{N}(\mu_{\tau_i}, \sigma_{\tau_i})$, causing random walk and white frequency fluctuations, respectively. Possible nonzero mean values of such variables are responsible for systematic linear frequency and time drifts, respectively. It is worth emphasizing that the assumption of normally distributed increments for the state variables is in agreement with the most sophisticated clock models available in the literature such as in [15]. However, as stated in Section I, only two state variables are used in (1), because the dynamic of clock acceleration/deceleration due to aging is negligible over short time intervals.
4) $\mathbf{s}_i(t)$ is a further input modeling the influence of different environmental factors (e.g., temperature or vibrations) on the clock rate.
5) $\mathbf{u}_i(t) = [u_{\rho_i}(t), u_{\tau_i}(t)]^T$ is the potential control action on both the clock offset and the clock rate if the clock is disciplined by a controller (otherwise this input is 0).
6) $c_i(t)$ is the system output, namely the timestamp measured by clock i anytime a message is sent or received by node i.
7) $\epsilon_i(t)$ represents the total timestamping error given by the superimposition of multiple systematic and random contributions such as finite clock resolution, oscillator white phase noise, jitter, and delays introduced at the MAC and/or at the physical layer anytime a message is sent or received at time t. It is important to highlight that the values of $\epsilon_i(t)$ generally differ when a message is sent or received. In particular, the receiving timestamps are often affected by a larger uncertainty because of the additional time required by the transceiver for symbol synchronization, i.e., to detect when the frame actually begins.

Also, as explained in Section II, the time spent by the various synchronization frames to cross switches and connections between nodes must be properly estimated and compensated. If $b_j(k)$ is the *bridge delay* when switch j is crossed by the kth *Sync* message and $l_{ij}(k)$ denotes the corresponding one-hop *line delay* between nodes j and i, the total communication latency between the synchronization master (in the following denoted with index 0) and one of the input ports of node i is given by

$$\delta_{i0}(k) = b_j(k) + l_{ij}(k) + \delta_{j0}(k) \tag{2}$$

with $\delta_{00}(k) = 0$ and $b_0(k) = 0\ \forall k$ by definition. In practice, $b_j(k)$ may exhibit significant random fluctuations due to message buffering, priority-related issues, variable traffic conditions and frame size. On the contrary, $l_{ij}(k)$ is quite deterministic if the network topology is fixed, but $l_{ij}(k)$ can differ from $l_{ji}(k)$ because of cable skewness and other asymmetries at the physical layer. It has to be noted that (2) can be used iteratively, to compute all the delays at the input ports throughout the network. Therefore, in Section IV the technique to estimate the individual values of $b_j(k)$ and $l_{ij}(k)$ is described.

IV. Estimators Definition

System (1) describes a continuous-time clock model. However, node synchronization occurs only at discrete times, i.e., when a *Sync* frame is actually received by some node. This means that the synchronization model is event-based and should be discretized in time with a nominal period T_s. Assume, for the sake of simplicity, but without loss of generality, that a linear path rooted in the grandmaster, composed by L nodes (with $L \leq N$) and ending in a leaf of the tree is extracted from the network shown in Fig. 1. Anytime a message has to be processed by some node, various types of data are required to identify univocally the instant at which this event occurs:

1) the timescale on which the time of the event is measured;
2) whether the message is sent or received;
3) the identification number of the node transmitting or receiving the message;
4) the sequence number of the message to be processed.

Since in (1) t denotes the ideal timescale, in the following we will refer to $\bar{t}^{\mathrm{s}}_{i_k}$, $\underline{t}^{\mathrm{s}}_{i_k}$ as the instants (on the ideal timescale) when the kth *Sync* frame is sent or received, respectively, by node i. Similarly, let $\bar{t}^{\mathrm{d}}_{i_m}$, $\underline{t}^{\mathrm{d}}_{i_m}$ $\bar{t}^{\mathrm{r}}_{i_m}$, and $\underline{t}^{\mathrm{r}}_{i_m}$ be the instants when the mth *Delay_Req* and *Delay_Res* frames are either sent or received by node i, respectively. Generally, $k \neq m$ because $T_{dq} \leq T_s$, as explained in Section II. In this respect, it is worth emphasizing that the synchronization period T_s changes over time and in different points of the network, because it depends on the position of node i, as well as on data traffic and protocol-related random communication latencies. Hence, the actual duration (on an ideal timescale) of the kth synchronization period on node i is $T_{s_{i_k}} = \underline{t}^{\mathrm{s}}_{i_{k+1}} - \underline{t}^{\mathrm{s}}_{i_k}$.

If $\mathbf{e}_i(k) = \mathbf{x}_i(\underline{t}^{\mathrm{s}}_{i_k}) - \mathbf{x}_0(\underline{t}^{\mathrm{s}}_{i_k})$ represents the difference between the state of clock i and the state of the synchronization master when the kth timestamp is received by clock i, and if the control input $\mathbf{u}_i(k)$ is constant between subsequent *Sync* frames (with $\mathbf{u}_0(k) = 0$ by definition), it can be easily proved that the clock error model associated to (1) and discretized with period $T_{s_{i_k}}$ is given by [26]

$$\begin{aligned}\mathbf{e}_i(k+1) &= F_i(k)\mathbf{e}_i(k) + G_i(k)\mathbf{u}_i(k) + \mathbf{q}_i(k) + \mathbf{v}_i(k)\\ y_i(k) &= C\mathbf{e}_i(k) + w_i(k)\end{aligned} \tag{3}$$

where

$$F_i(k) = e^{(\underline{t}^{\mathrm{s}}_{i_{k+1}} - \underline{t}^{\mathrm{s}}_{i_k})A} \tag{4}$$

$$G_i(k) = \int_{\underline{t}^{s}_{i_k}}^{\underline{t}^{s}_{i_{k+1}}} e^{(\underline{t}^{s}_{i_{k+1}}-t)A}dt \tag{5}$$

$$\mathbf{q}_i(k) = \int_{\underline{t}^{s}_{i_k}}^{\underline{t}^{s}_{i_{k+1}}} e^{(\underline{t}^{s}_{i_{k+1}}-t)A}B[\mathbf{s}_i(t)-\mathbf{s}_0(t)]dt \tag{6}$$

$$\mathbf{v}_i(k) = \int_{\underline{t}^{s}_{i_k}}^{\underline{t}^{s}_{i_{k+1}}} e^{(\underline{t}^{s}_{i_{k+1}}-t)A}[\mathbf{n}_i(t)-\mathbf{n}_0(t)]dt \tag{7}$$

$y_i(k) = c_i(\underline{t}^{s}_{i_k}) - c_0(\underline{t}^{s}_{i_k})$ and $w_i(k) = \epsilon_i(\underline{t}^{s}_{i_k}) - \epsilon_0(\underline{t}^{s}_{i_k})$. Notice that while $c_i(\underline{t}^{s}_{i_k})$ is read directly from node i, $c_0(\underline{t}^{s}_{i_k})$ cannot be measured directly, because $\underline{t}^{s}_{i_k}$ refers to the instant when the *Sync* message is received by node i, which is unknown to the master. However, since $\underline{t}^{s}_{i_k} = \bar{t}^{s}_{0_k} + \delta_{i0}(k)$, then

$$c_0\left(\underline{t}^{s}_{i_k}\right) = c_0\left(\bar{t}^{s}_{0_k}\right) + \delta_{i0}(k) + \epsilon_0\left(\bar{t}^{s}_{0_k}\right). \tag{8}$$

In practice, $\delta_{i0}(k)$ has to be estimated on the timescale of the master clock. In the rest of this paper, the *hat* symbol and the i superscript are used to denote that a certain quantity is estimated by node i. For instance, the total communication delay from node 0 to node i estimated on the master timescale is derived from (2)

$$\hat{\delta}^{0}_{i0}(k) = \hat{b}^{0}_{j}(k) + \hat{l}^{0}_{ij}(m) + \hat{\delta}^{0}_{j0}(k) \tag{9}$$

where, without loss of generality, j in this case refers to the node just preceding node i in the chain and

$$\hat{b}^{0}_{j}(k) = \left[c_j\left(\bar{t}^{s}_{j_k}\right) - c_j\left(\underline{t}^{s}_{j_k}\right)\right]\frac{\hat{\rho}_0\left(\underline{t}^{s}_{j_k}\right)}{\hat{\rho}_j\left(\underline{t}^{s}_{j_k}\right)} = \frac{c_j\left(\bar{t}^{s}_{j_k}\right) - c_j\left(\underline{t}^{s}_{j_k}\right)}{\hat{\rho}_j\left(\underline{t}^{s}_{j_k}\right)} \tag{10}$$

is the estimated bridge delay of node j resulting from the difference of the egress-ingress timestamps associated to the kth *Sync* frame, multiplied by the *rate compensation factor* given by the ratio between the rate of the master and the estimated rate of clock j. Consider that all the clock rate estimates are initialized to 1, but they are generally different from 1 because of clock nonidealities. The value of $\hat{\rho}_0(t)$ instead, is identically equal to 1 by definition because node 0 is the synchronization master of the network.

Similarly, under the assumption that only peer-to-peer TCs are used, the *line delay* between nodes j and i on the master timescale is given by

$$\hat{l}^{0}_{ij}(m) = \hat{l}^{i}_{ij}(m)\frac{\hat{\rho}_0(\underline{t}^{r}_{i_m})}{\hat{\rho}_i(\underline{t}^{r}_{i_m})} = \frac{\hat{l}^{i}_{ij}(m)}{\hat{\rho}_i(\underline{t}^{r}_{i_m})} \tag{11}$$

where

$$\hat{l}^{i}_{ij}(m) = \frac{c_i\left(\underline{t}^{r}_{i_m}\right) - c_i\left(\bar{t}^{d}_{i_m}\right) - \left[c_j\left(\bar{t}^{r}_{j_m}\right) - c_j\left(\underline{t}^{d}_{j_m}\right)\right]\frac{\hat{\rho}_i\left(\underline{t}^{r}_{i_m}\right)}{\hat{\rho}_j\left(\bar{t}^{r}_{i_m}\right)}}{2} \tag{12}$$

is the *line delay* estimated by node i on its own timescale by halving the RTT that results from the exchange of the mth *Delay_Req/Delay_Res* message pair. Notice that the rightmost term at the numerator of (12) represents the *rate compensation factor* between nodes i and j and it is used to map the timestamp values measured by node j on the timescale of node i. Of course, as noted previously, $\hat{\delta}^{0}_{j0}(k)$ in (9) is computed by applying iteratively expressions (10)–(12).

If the random variable $\eta_{0i}(k)$ is supposed to include the uncertainty contributions associated with the measurement both of $\hat{\delta}^{0}_{j0}(k)$ and $\hat{l}^{0}_{ij}(m)$, then (8) can be also rewritten as

$$c_0\left(\underline{t}^{s}_{i_k}\right) = c_0\left(\bar{t}^{s}_{0_k}\right) + \hat{\delta}^{0}_{i0}(k) + \epsilon_0\left(\bar{t}^{s}_{0_k}\right) + \eta_{0i}(k) \tag{13}$$

and the measurement uncertainty term $w_i(k)$ in (3) has to be replaced by $\tilde{w}_i(k) = w_i(k) + \eta_{0i}(k)$. Therefore, if i) the influence of the environmental factors is negligible (i.e., $\mathbf{q}_i(k) \approx 0$) and ii) the uncertainty contributions $\tilde{w}_i(k)$ and $\mathbf{v}_i(k)$ are weakly correlated, then the state of (3) can be estimated by a KF, even if the KF can be hardly regarded as optimal in the case considered [12]. In particular, the *prediction* equations are

$$\begin{aligned}\hat{\mathbf{e}}^{+}_{i}(k+1) &= F_i(k)\hat{\mathbf{e}}_i(k) + G_i(k)\mathbf{u}_i(k)\\ \hat{y}^{+}_{i}(k+1) &= C\mathbf{e}^{+}_{i}(k+1)\\ P^{+}_{i}(k+1) &= F_i(k)P_i(k)F^{T}_{i}(k) + Q_i(k)\end{aligned} \tag{14}$$

where $\mathbf{e}^{+}_{i}(k+1)$ is the predicted state, $\hat{y}^{+}_{i}(k+1)$ is the predicted output vector, $P_i(k)$ and $P^{+}_{i}(k+1)$ are the estimated and predicted state covariance matrices, respectively, and $Q_i(k)$ is the covariance matrix of $\mathbf{v}_i(k)$, given by

$$Q_i(k) = \int_{\underline{t}^{s}_{i_k}}^{\underline{t}^{s}_{i_{k+1}}} e^{(\underline{t}^{s}_{i_{k+1}}-t)A}\begin{bmatrix}\sigma^2_{\rho_i} & 0\\ 0 & \sigma^2_{\tau_i}\end{bmatrix}e^{(\underline{t}^{s}_{i_{k+1}}-t)A^T}dt \tag{15}$$

where σ_{ρ_i} and σ_{τ_i} are the standard deviations of the continuous-time noise terms in $\mathbf{n}_i(t)$ as defined in (1). Dually, the *update* equations of the KF are

$$\begin{aligned}\hat{\mathbf{e}}_i(k+1) &= \mathbf{e}^{+}_{i}(k+1) + \bar{K}_i(k)[y_i(k+1) - \hat{y}_i(k+1)]\\ P_i(k+1) &= [I_2 - \bar{K}_i(k+1)C]P^{+}_{i}(k+1)\end{aligned} \tag{16}$$

where

$$\bar{K}_i(k+1) = P^{+}_{i}(k+1)C^T[CP^{+}_{i}(k+1)C^T + D_i(k+1)]^{-1} \tag{17}$$

is the *Kalman gain* and $D_i(k+1)$ is the variance of $\tilde{w}_i(k+1)$, which results from the sum of the variance of $w_i(k+1)$ due to the overall timestamping uncertainty [as defined in (8)] and of the variance of $\eta_{0i}(k+1)$ associated to the measurement of bridge and line delays up to the ith node [according to (2)]. It is worth emphasizing that the proposed KF works also under the effect of sudden changes of the environmental quantities [i.e., when $\mathbf{q}_i(k) \neq 0$], provided that a suitable *fudge factor* is introduced to handle such events [13].

V. Servo-Clock Design

Although servo-clocks are not always strictly required for TC implementation, the use of a controller to discipline the local clock is often welcome and useful in practice [27]. In particular, the clock state resulting from the KF can be used not only to compute the *rate compensation factors* for *bridge delay* and *line delay* estimation [28] [e.g., from (10) and (11)], but also to drive an LQR controller, which makes the elements of $\mathbf{e}_i(\cdot)$ converge asymptotically to zero as $k \to +\infty$, with uncertainty given by the KF. While, as stated in Section I, other solutions proposed in the literature usually assume that the synchronization period is fixed and known [16], T_s may actually fluctuate as described in Section III. For instance,

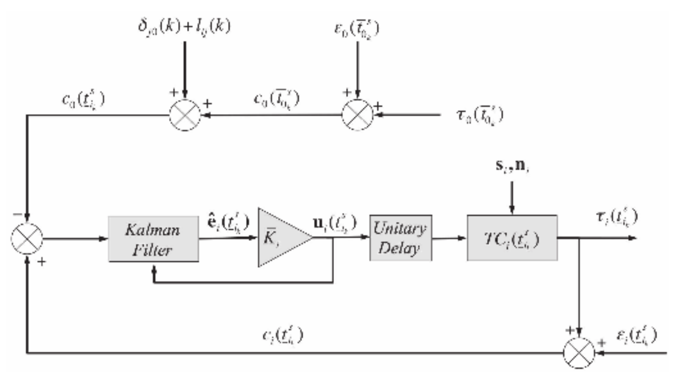

$\delta_{y0}(k)+l_{ij}(k)$
$\varepsilon_0(\bar{t}_{0_k}')$
$c_0(t_{i_k}^s)$
$c_0(\bar{t}_{0_k}')$
$\tau_0(\bar{t}_{0_k}')$
$\mathbf{s}_i, \mathbf{n}_i$
Kalman Filter
$\hat{\mathbf{e}}_i(t_{i_k}')$
$\bar{K}_i$
$\mathbf{u}_i(t_{i_k}^s)$
Unitary Delay
$TC_i(t_{i_k}')$
$\tau_i(t_{i_k}^s)$
$c_i(t_{i_k}')$
$\varepsilon_i(t_{i_k}')$

Fig. 2. Control loop model for the *i*th disciplined TC.

according to the Standard IEEE 1588:2008, the time interval between two consecutive *Sync* frames is allowed to change within ±30% of the nominal value with 90% confidence [9]. This phenomenon is of primary importance for control design, since such a time variability could lead to instability or, at least, to poor closed-loop performance [17]. Moreover, the control action cannot be applied on an ideal timescale, since every controller can rely only on the timescale of the free-running TC. As a consequence, a precise analysis of the control model is needed.

A. Control Model Definition

Due to the *separation principle*, the controller designed in this paper can rely on the combination of the KF and a static-state feedback

$$\mathbf{u}_i(k) = R_i \hat{\mathbf{e}}_i(k-1) \tag{18}$$

where $R_i \in \mathbb{R}^{2\times 2}$. This is the classic approach used also for linear quadratic Gaussian (LQG) control. In the following, the feedback matrix R_i is explicitly designed to consider possible T_s fluctuations, while assuring *mean square stability* with a minimum variance of the clock state. If the variable $S_i(k)$ denotes the covariance matrix of the controlled clock state when the kth *Sync* frame is received, the closed-loop system is referred to as *mean square stable* if $\lim_{k\to\infty} S_i(k) = \overline{S}_i < +\infty$ [29]. Clearly, the smaller the value of $\overline{S}_i$, the better the controller. Therefore, $\overline{S}_i$ can be regarded as a metric of QoC.

Fig. 2 shows the whole closed-loop system, along with the corresponding uncertainty sources as inputs. Notice that in (18) the control action is applied after a *unitary delay* element corresponding to one synchronization period. This is due to the fact that the processing time spent to estimate the clock state in the KF is not negligible. Therefore, to limit the jitter in applying the control, $\mathbf{u}_i(k)$ is kept constant and the new value is applied only when the next synchronization event occurs, as customary in digital control systems. It is worth emphasizing that the delayed application of the control input comes in handy if a two-step clock synchronization model is used.

To analyze performance and stability of the closed-loop system, an additional serious practical issue should be considered when the controller is implemented. In fact, the duration $T_{s_{i_k}}$ of the kth synchronization period on node i is actually measured by the ith clock itself. Therefore, the corresponding measurement result $T^i_{s_{i_k}}$ (where, again, the superscript i represents the timescale on which the time interval is measured) depends not only on the intrinsic variability of T_s, but also on the limited resolution and the frequency fluctuations of the local clock. Accordingly, we can write that $T^i_{s_{i_k}} = \frac{d_{i_k}}{f_i}$, where f_i is the nominal frequency of the ith clock and d_{i_k} in the integer number of ticks counted in the kth synchronization period. While the value of d_{i_k} should be ideally equal to a given value M, in practice it generally lies within the tolerance interval $\mathcal{M} = [M_l, M_u]$, whose upper and lower bounds depend on the chosen synchronization protocol. If d_{i_k} is out of this interval, most probably the kth *Sync* frame is lost. This implies that the controller cannot be updated and the previous value must be kept for a further synchronization interval. As a consequence of the assumptions above, if the general clock model (1) is discretized with a period $\frac{1}{f_i}$, the system and input matrices of the discretized clock error model result from expressions similar to (4) and (5):

$$\tilde{F}_i = \begin{bmatrix} 1 & 0 \\ \frac{1}{f_i} & 1 \end{bmatrix} \text{ and } \tilde{G}_i = \begin{bmatrix} \frac{1}{f_i} & 0 \\ \frac{1}{2f_i^2} & \frac{1}{f_i} \end{bmatrix}. \tag{19}$$

Notice that $\tilde{F}_i$ and $\tilde{G}_i$ are time invariant. However, the variability of the *Sync* frame interarrival times causes the closed-loop system switching between different dynamics that depend on the actual value of d_{i_k}. Thus, the closed-loop dynamic can be modeled as

$$\mathbf{z}_i(k+1) = \begin{cases} A_{d_{i_k}}\mathbf{z}_i(k) + \tilde{B}\mathbf{v}_i(k) + \tilde{E}\boldsymbol{\gamma}_i(k) & d_{i_k} \in \mathcal{M} \\ A_{o_i}\mathbf{z}_i(k) + \tilde{B}\mathbf{v}_i(k) + \tilde{E}\boldsymbol{\gamma}_i(k) & d_{i_k} \notin \mathcal{M} \end{cases} \tag{20}$$

where $\mathbf{z}_i(k) = [\mathbf{e}_i(k)^T, \mathbf{u}_i(k)^T]^T$, $\mathbf{v}_i(k)$ is the same as in (3), $\boldsymbol{\gamma}_i(k)$ is the column vector containing the estimation fluctuations associated with $\hat{\mathbf{e}}_i(k)$ at the output of the KF

$$A_{d_{i_k}} = \begin{bmatrix} \tilde{F}_i^{d_{i_k}} & \tilde{G}_{d_{i_k}} \\ R_i & 0_2 \end{bmatrix}, \quad A_{o_i} = \begin{bmatrix} \tilde{F}_i^M & \tilde{G}_M \\ 0_2 & I_2 \end{bmatrix} \tag{21}$$

are the closed system matrices depending on whether a *Sync* frame is received or not, I_2 and 0_2 are the 2×2 identity and all-zeros matrix, respectively, and

$$\tilde{B} = \begin{bmatrix} I_2 \\ 0_2 \end{bmatrix}, \quad \tilde{E} = \begin{bmatrix} 0_2 \\ R_i \end{bmatrix} \tag{22}$$

represent the input matrices of the closed-loop system. Observe that matrix $\tilde{G}_{d_{i_k}} = \sum_{j=0}^{d_{i_k}-1} \tilde{F}_i^{d_{i_k}-j-1}\tilde{G}_i$ in (21) is built as a result of the computation of the forced response of the system when a constant control input $\mathbf{u}_i(k)$ is applied. Also, $\tilde{G}_M$ is simply the same as $\tilde{G}_{d_{i_k}}$ when $d_{i_k} = M$.

B. Controller Analysis and Synthesis

The stochastic process generating the synchronization period is considered here to be stationary in time. This assumption can be removed at the cost of a more complicated model, which will be the subject of a future work. Under the assumption of stationarity, the probability μ_{d_i} that the measured synchronization period on node i occurs after $d_i \in \mathcal{M}$ ticks is independent of k. Therefore, the probability of not receiving any *Sync* frame on node i is simply given by $\mu_{o_i} = 1 - \sum_{d_i \in \mathcal{M}} \mu_{d_i}$. Of course, μ_{o_i} depends on the position of node i in the network. In particular, μ_{o_i} typically grows as i is closer to the end of a chain.

If $S_i(k)$ is the covariance matrix of $\mathbf{z}_i(k)$, its value after the $(k+1)$th synchronization period is given by

$$S_i(k+1) = \mathsf{E}\left\{\mathbf{z}_i(k+1)\mathbf{z}_i(k+1)^T\right\}$$
$$= \sum_{d_i=M_l}^{M_u} \mu_{d_i} A_{d_i} S_i(k) A_{d_i}^T + \mu_{o_i} A_{o_i} S_i(k) A_{o_i}^T + H_i(k) \qquad (23)$$

where $H_i(k)$ is the total covariance matrix associated to $[\mathbf{v}_i(k)^T, \boldsymbol{\gamma}_i(k)^T]^T$ that includes $Q_i(k)$ and $R_i P_i(k) R_i^T$ on the main diagonal. If operator $\otimes$ denotes the Kronecker product, by using the operator's properties, (23) can be rewritten as

$$\mathrm{vec}(S_i(k+1)) = \Gamma_A \mathrm{vec}(S_i(k)) + \mathrm{vec}(H_i(k)) \qquad (24)$$

where operator vec$(\cdot)$ builds a vector by stacking the columns of the input matrix and

$$\Gamma_{A_i} = \sum_{d_i \in \mathcal{M}} \mu_{d_i}(A_{d_i} \otimes A_{d_i}) + \mu_{o_i}(A_{o_i} \otimes A_{o_i}). \qquad (25)$$

Observe that the discrete-time time-invariant system in (24) can be regarded as a standard linear system with state vec$(S_i(k))$ and input vec$(H(k))$. If the positive-definite covariance matrix $H(k)$ is bounded (i.e., $H(k) \leq \overline{H}$), the *mean square stability* is guaranteed if and only if $S_i(k)$ in (23) tends toward a finite value for $k \to +\infty$, which occurs when

$$\zeta(\Gamma_{A_i}) = \max_j |\lambda_j\left(\Gamma_{A_i}\right)| < 1 \qquad (26)$$

where $\lambda_j(\Gamma_{A_i})$ denotes the jth eigenvalue of Γ_{A_i}. In such conditions, system (24) is asymptotically stable and it admits a steady-state solution given by

$$\mathrm{vec}(\overline{S}_i) = (I_{16} - \Gamma_{A_i})^{-1}\mathrm{vec}(\overline{H}). \qquad (27)$$

In conclusion, if the static feedback controller (18) is used to discipline the clocks (for instance, using an LQG synthesis) and condition (26) is satisfied, the closed-loop system is stable even under the effect of synchronization period fluctuations within $\mathcal{M}$. Moreover, the trace or the determinant of $\overline{S}_i$ can be used to determine the expected variance of the controlled variables. Clearly, the same expression can be used to synthesize the controller. To this purpose, it is sufficient to find the elements of the static feedback matrix R_i such that the determinant of $\overline{S}_i$ is minimized and constraint (26) is met. Although, in principle, R_i could be a row vector since the system is completely controllable by driving the clock rate, in practice the optimization problem above may hardly assure good performances. In fact, it is better to control both time and clock rate. However, in this case R_i turns to be a square bi-dimensional matrix (i.e., with up to four parameters), which makes the corresponding nonlinear optimization problem very hard to solve. Therefore, we finally decided to set R_i as a diagonal matrix (with two design parameters only), because this approach assures a good tradeoff between flexibility and optimization complexity. As a consequence, it can be shown that the elements of the main diagonal have to be drawn from the open compact set $(-f_i/M, 0)$. In this way, at first all the values satisfying (26) are found numerically. Then, the solution minimizing (27) is chosen. It is worth noticing that the increment in complexity required to solve the full four-variable optimization problem is probably excessive with respect to the performance improvement achievable using just a diagonal matrix.

VI. Performance Evaluation: A Case Study

To evaluate the performances of the proposed clock synchronization strategy in a realistic scenario, several simulations have been performed in Matlab assuming to run PTCP in a PROFINET IO network. In the following, at first the main features of PROFINET IO are shortly recalled. Then, the values of the various simulation parameters are introduced and justified. Finally, the corresponding results are reported.

A. Protocol Overview

PROFINET IO is an industrial automation protocol for exchanging data between IO-controllers (i.e., intelligent devices running automation control programs) and IO-devices (i.e., sensors, actuators and IO modules). PROFINET IO is defined in the communication profile specifications CP 3/4, CP 3/5 and CP 3/6 of the Standard IEC 61784-2 [30]. PROFINET IO comprises several combinations of features and parameters. The features considered for the simulations of this paper refer to the so-called PROFINET IO Conformance Class C (CC-C). In this class, process/field data are exchanged cyclically between IO-controller and IO-devices using the *RT_Class* 3 communication protocol [also known as isochronous real-time (IRT)]. Usually, such a high-performance protocol is used when the application (e.g., motion control) needs a cycle time in the range of 31.25 μs to 4 ms with extremely low jitter (± 1 μs). The RT_Class 3 devices require synchronous communication based on an ad-hoc time division medium access (TDMA) policy. In particular, medium access is periodic and every cycle time is divided into two main phases.

1) A RESERVED phase, in which the whole network infrastructure is used to transmit IRT frames only. Other Transport Control Protocol/Internet Protocol (TCP/IP) or PROFINET IO messages just wait in the buffers of the network switches, while the IRT time-critical frames are scheduled *a priori* and routed over predefined paths.
2) An OPEN or GREEN phase that is used for non-time-critical PROFINET and Ethernet data traffic, including usual IP/TCP frames. The OPEN phase is always present in a cycle. During this phase, the frames are transmitted and routed according to their Ethernet priority (as specified in the Standard IEEE 802.1Q [31]).

Clock synchronization in PROFINET IO IRT networks can be advantageously exploited for automation applications, thus enabling high-end performance [32]. The topology of a PROFINET IO IRT network can be chosen in order to fulfill specific availability and maintenance needs. Typically,

TABLE I

CLOCK PARAMETER VALUES CHOSEN FOR THE SIMULATIONS. ALL THE NODES ARE ASSUMED TO BE NOMINALLY IDENTICAL

Parameter	Description	Value
f_i	Nominal clock frequency	100 MHz
$\rho_i(0)$	Initial value of the normalized clock rate	[1-25 ppm, 1+25 ppm]
μ_{ρ_i}	Mean value of the drift rate noise	0 ppm/s
σ_{ρ_i}	Standard deviation of the drift rate noise	10^{-2} ppm/s
$\tau_i(0)$	Initial value of the clock timers	$\tau_0(0) = 10$ s $\tau_i(0) = [7.5, 12.5]$ s
μ_{τ_i}	Mean value of the clock rate noise	0 ppm
σ_{τ_i}	Standard deviation of the clock rate noise	10^{-3} ppm

TABLE II

PTCP AND COMMUNICATION PARAMETER VALUES USED IN THE SIMULATIONS. ALL THE NODES ARE ASSUMED TO BE NOMINALLY IDENTICAL

Parameter	Description	Value
T_s	Synchronization period	$\mathcal{T}(29.97, 30.03)$ ms
b_i	Bridge delay	$\mathcal{B}(a, b)$ with $a = 10$ μs b=50,100,250 μs
T_{dq}	Nominal delay request period	8 s
T_d	*PTCP-DelayReqPDU* to *PTCP-DelayResPDU* delay	$\mathcal{U}(400, 800)$ μs
l_{ij}	Line delay	$\mathcal{U}(1602, 1608)$ ns
ϵ_i	TX timestamping RX timestamping	$\mathcal{D}(0, 2, 11, 13)$ ns $\mathcal{D}(3, 7, 13, 17)$ ns

the physical network topology is a ring in order to exploit redundancy of the double path, but logically it still behaves as a line (i.e., a chain of devices).

B. Simulation Parameters

The PROFINET IO IRT implementation includes PTCP for network-level clock synchronization. The simulation parameters can be roughly divided into two groups, i.e., those depending on the clock model and those related to PTCP. The former ones depend on the typical features of Ethernet clocks and are listed in Table I in accordance with the definitions reported in Sections III and IV (except the initial clock state values which are chosen just randomly). Table II, instead, shows the values of the chosen PTCP and communication parameters, as well as the corresponding uncertainty contributions. Some values are based on PROFINET IO specifications, while others are obtained experimentally, as described in [33], [34]. In Table II, $\mathcal{U}(a, b)$ and $\mathcal{T}(a, b)$ represent uniform and symmetric triangular distributions, respectively, with support in $[a, b]$. Symbol $\mathcal{D}(a, b, c, d)$ denotes a trapezoidal distribution with longer and shorter bases given by $d-a$ and $c-b$, respectively. Finally, $\mathcal{B}(a, b)$ denotes a Beta distribution defined in $[a, b]$, with parameters $\alpha = 1$ and $\beta = 3$. The nominal clock synchronization interval T_s is set equal to the PROFINET default value (i.e., 30 ms). Synchronization interval jitter is in the order of some tens of μs and exhibits a triangular distribution [33], [34]. The bridge delay values greatly depend on data traffic as well as on frame size. In PROFINET IO IRT, the *Sync* frames are forwarded (with a cut-through approach) with the highest priority only in the green phase. During the red (reserved) phase instead, the *Sync* frames wait inside the switch buffer till the beginning of the next green phase. On the basis of these assumptions, by extrapolating the experimental results reported in [33], [34], a Beta distribution is used to model the bridge delays. Observe that, while the minimum value $a = 10$ μs is constant (since it depends only on the time spent in crossing a switch when a *Sync* frame is forwarded immediately), different upper bounds are considered in simulations (i.e., $b = 50, 100, 250$ μs) to describe the effect of a growing amount of buffered data.

PROFINET IO relies on burst message exchange for line delay estimation. In particular, bursts of five *Delay_Req/Delay_Res* frames are issued with a nominal period $T_{dq} = 8$ s. The time distance between pairs of consecutive *Delay_Req* frames of the same burst is 200 ms. The latency T_d between a *Delay_Req* frame and the corresponding *Delay_Res* frame is uniformly distributed between 400 μs and 800 μs. Thus, each line delay value is obtained from the average of seven consecutive measurement results given by (11) (i.e., five belonging to the current burst plus two taken from the previous burst). All line delay values are assumed to be uniformly distributed in [1602, 1608] ns, as they include both cable and physical layer latencies. Finally, the hardware timestamping uncertainty contributions reported at the bottom of Table II are assumed to have approximately a trapezoidal distribution due to the superimposition of clock resolution and oscillator phase noise. This kind of distribution was obtained through Monte Carlo simulations using the individual uncertainty contributions reported in [13]. On the receiver side, slightly larger systematic delays and jitter are introduced by clock signal recovery circuitry. Finally, the nominal probability of losing a packet has been evenly fixed to 0.2%. However, this probability grows with the distance of the node from the master clock, as discussed in Section V.

C. Simulation Results

The simulation results reported in this section refer to a linear network topology with at most 30 nodes between the synchronization master and the end of the line, in accordance with PROFINET IO specifications. Every Monte Carlo simulated experiment consists of 30 runs. Each test lasts 60 s. The root mean square estimation errors (RMSEs) of time and rate offsets of six controlled network nodes (i.e., nodes 1, 2, 5, 10, 20, and 30) with respect to the master are plotted as a function of time in Fig. 3. All *bridge delays* are distributed as $\mathcal{B}(10\mu\text{s}, 50\mu\text{s})$. In particular, Fig. 3(a) and (b) shows the results for a controller that stabilizes each clock node and minimizes the closed-loop maximum eigenvalue, hence ensuring the fastest convergence. As a consequence, the transient is very short (less than 3 s), but the system tends to amplify the uncertainties associated with master timestamp propagation. On the other hand, Fig. 3(c) and (d) reports the

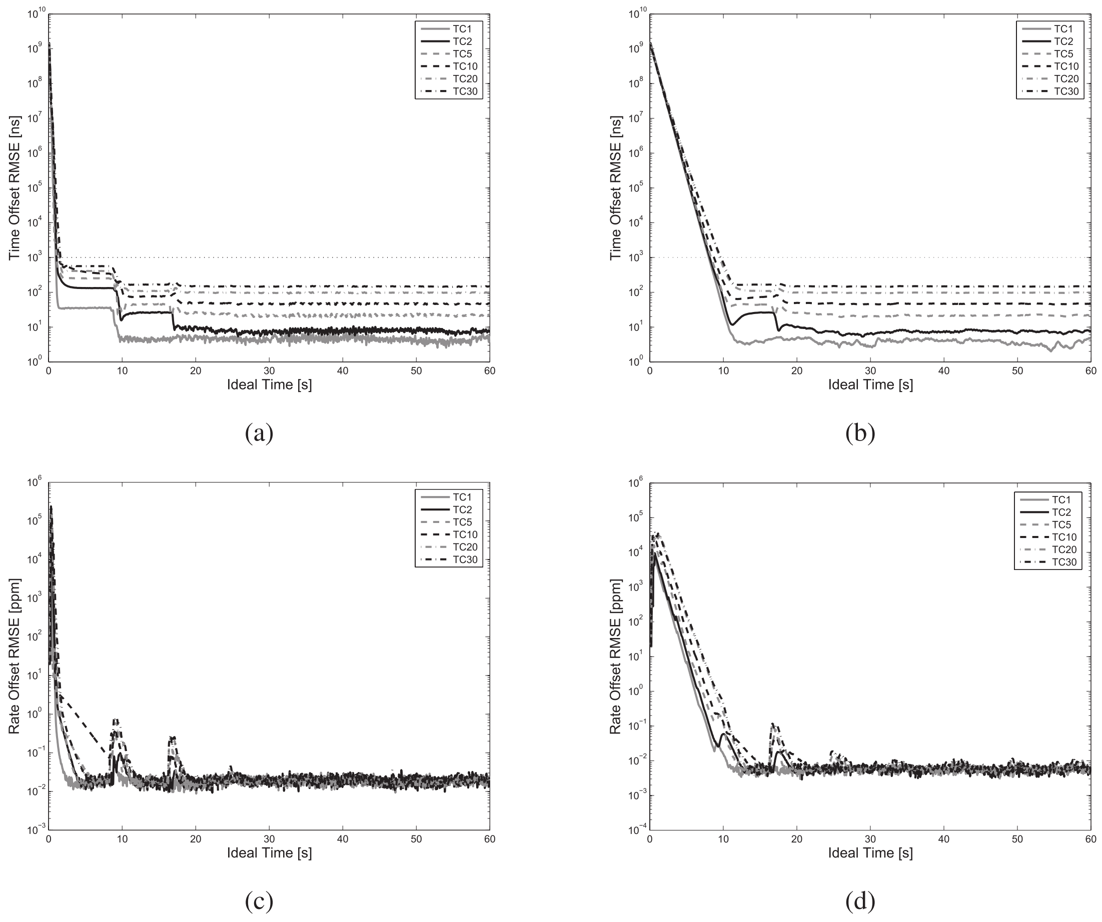


Fig. 3. (a)–(d) RMSE values of the time offsets [(a) and (b)] and of the rate skews [(c) and (d)] of some nodes of the chain with respect to the master for two different settings of the designed servo-clock parameters, i.e., fastest convergence (on the left) and minimum determinant of the closed-loop steady-state covariance (on the right). All the *bridge delays* are assumed to be distributed as $\mathcal{B}(10\mu\text{s}, 50\mu\text{s})$. In (a) and (b), the 1 μs upper bound to synchronization accuracy (typical in industrial automation) is represented by a dotted line.

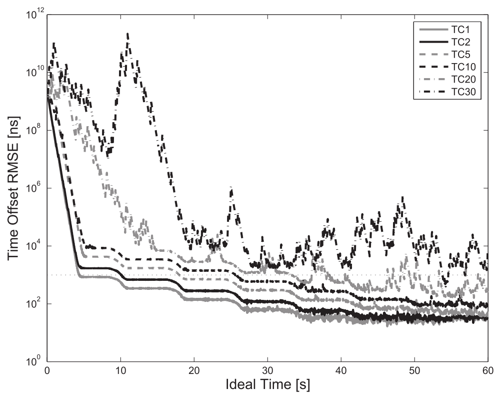


Fig. 4. RMSE values of the time offsets of some nodes with respect to the master obtained using a typical PI-based servo-clock. The *bridge delays* is again assumed to be distributed as $\mathcal{B}(10\mu\text{s}, 50\mu\text{s})$. Horizontal dotted line: 1 μs upper bound to synchronization accuracy, which is typical in industrial automation.

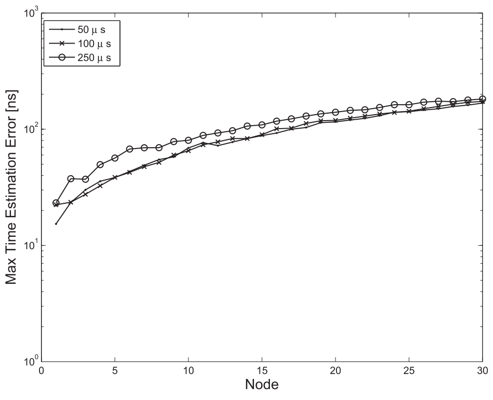
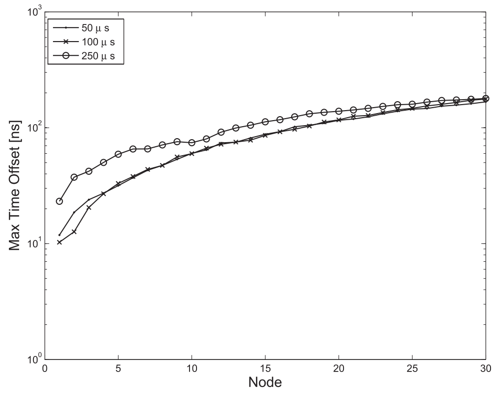
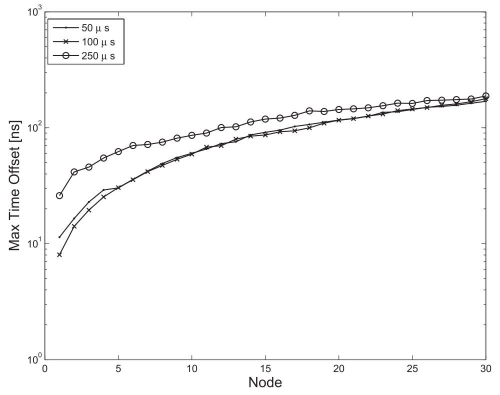


Fig. 5. Maximum time offset estimation errors for a cascade of 30 TCs after 60 s. Different lines refer to three different *bridge delay* models, i.e., $\mathcal{B}(10\ \mu\text{s}, 50\ \mu\text{s})$, $\mathcal{B}(10\ \mu\text{s}, 100\ \mu\text{s})$, and $\mathcal{B}(10\ \mu\text{s}, 250\ \mu\text{s})$.

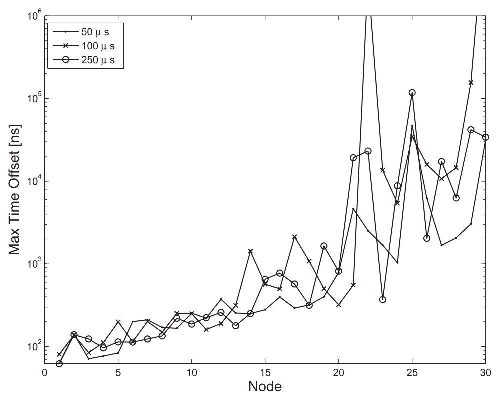
same types of errors obtained with a controller that minimizes the determinant of the steady-state output covariance $\overline{S}_i$. In this way, a filtered, but slower, response is obtained. Observe that the RMSE values of the time offsets after reaching the steady state are below 200 ns, even at node 30. This value is much smaller the typical time offset of 1 μs required by top applications in industrial automation [1], [4] and represented by a dotted horizontal line in Fig. 3(a) and (b). Clearly, the clock state estimation accuracy is strongly influenced by line delay compensation, which is quite poor during the



(a)



(b)



(c)

Fig. 6. (a)–(c) Maximum time offsets between the servo-clocks and the synchronization master after 60 s. Different lines refer to different *bridge delay* models, i.e., $\mathcal{B}(10\ \mu\text{s}, 50\ \mu\text{s})$, $\mathcal{B}(10\ \mu\text{s}, 100\ \mu\text{s})$, and $\mathcal{B}(10\ \mu\text{s}, 250\ \mu\text{s})$. The patterns in (a) and (b) refer to the same two controllers minimizing the transient length and the determinant of the closed-loop state covariance matrix, respectively. The curves in (c) instead are obtained using the same PI controller as in Fig. 4.

transient phase. However, accuracy improves every 8 s, i.e., anytime a new burst of *Delay_Req/Delay_Res* frames is exchanged between pairs of consecutive nodes. Once the steady-state is reached, the line delay estimation uncertainty is negligible. Consider that (26) is a necessary and sufficient condition for mean square stability. Therefore, a static controller that does not explicitly take into account the fluctuations of the synchronization period or the possibility of losing some packets may exhibit poor performance or even instability. In order to highlight this issue, some simulation results performed in the same conditions as in Fig. 3, but based on a PI controller adjusting the time increments of each clock are shown in Fig. 4. Also in this case the dotted horizontal line represents the time offset of 1 μs typically required in industrial automation. The case of PI controllers is perfectly suitable for a comparison with the proposed approach, because such controllers are simple and commonly employed in servo-clocks [35]. In particular, the chosen PI-based servo-clock implementation is similar to the solution described in [36], but relies on a different set of control coefficients (i.e., $K_p = 48.7805$ and $K_I = 30.5$) that ensure a very short transient if no synchronization period jitter is present. In Fig. 4, the RMSE patterns of the time offsets as a function of time are shown. Quite interestingly, the steady-state synchronization accuracy on the first nodes of the chain are stable and just slightly worse than the results shown in Fig. 3(a) and (b). However, the clocks that are farther from the master (e.g., nodes 20 and 30) exhibit increasingly large time offset fluctuations due to the accumulated jitter which is not properly handled by the PI controller. Moreover, even a slight change of the PI coefficients or a small jitter increment easily leads to instability, as it will be shown in the following.

Fig. 5 shows the maximum time offset estimation errors associated with each KF after 60 s. The error patterns are shown as a function of the position of each node in the line (the master being node 0) and for different distributions of the bridge delays. The distance between pairs of consecutive nodes is assumed to be the same in all cases.

As expected, the estimation uncertainty grows as the distance from the master clock increases. This effect, in the worst case, is more evident when the bridge delays fluctuations are larger. Nevertheless, the uncertainty growth rate is generally compatible with PROFINET IO IRT requirements [32]. It is interesting to compare these results with the accuracy of the controlled clock with respect to the reference master time. Indeed, the worst-case accuracy of the "fastest convergence" controller [Fig. 6(a)] and of the "minimum closed-loop covariance" [Fig. 6(b)] controller are tightly related to the accuracy of the estimation process.

The situation is completely different when the PI-based servo-clock is used. In this case, if the bridge delay fluctuations increase, the maximum time offset oscillations clearly diverge with the position of the nodes in the chain of clocks, because the system is no longer stable.

VII. Conclusion

An accurate servo-clock model for chains of TCs is presented in this paper. The model explicitly considers noises and uncertainty contributions that usually affect industrial networks needing accurate clock synchronization. In addition, the model also considers some practical implementation issues. The main novelty of the proposed approach is its ability to tackle the detrimental effect of synchronization periods fluctuations on servo-clock performance. In this respect, a necessary and sufficient condition for mean square stability is given. With this condition, both a stability analysis and a controller design criterion are derived. Several simulations have been carried out in a case study based on the features of PROFINET IO devices to prove the effectiveness of the proposed approach in assuring a stable behavior over long chains of clocks, which cannot be guaranteed with other standard approaches. In particular, two solutions with different performances are presented: one exhibiting the fastest convergence time and the other minimizing the determinant of the closed-loop system state covariance matrix.

REFERENCES

[1] P. Ferrari, A. Flammini, D. Marioli, S. Rinaldi, E. Sisinni, A. Taroni, *et al.*, "Clock synchronization of PTP-based devices through PROFINET IO networks," in *Proc. Int. Conf. ETFA*, Sep. 2008, pp. 496–499.

[2] R. L. Scheiterer, C. Na, D. Obradovic, and G. Steindl, "Synchronization performance of the precision time protocol in industrial automation networks," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 6, pp. 1849–1857, Jun. 2009.

[3] J. L. B. Pedro V. Estrela, "Challenges deploying PTPv2 in a global financial company," in *Proc. Int. IEEE Symp. Precis. Clock Synchron. Meas., Control Commun.*, Sep. 2012, pp. 1–6.

[4] D. M. E. Ingram, P. Schaub, D. Campbell, and R. Taylor, "Evaluation of precision time synchronization methods for substation applications," in *Proc. IEEE ISPCS Meas., Control Commun.*, Sep. 2012, pp. 1–6.

[5] S. Meier, H. Weibel, and K. Weber, "IEEE 1588 syntonization and synchronization functions completely realized in hardware," in *Proc. ISPCS, Meas., Control Commun.*, Sep. 2008, pp. 1–4.

[6] A. Bondavalli, F. Brancati, A. Ceccarelli, and M. Vadursi, "Experimental validation of a synchronization uncertainty-aware software clock," in *Proc. IEEE Symp. Reliable Distrib. Syst.*, Oct./Nov. 2010, pp. 245–254.

[7] T. Cooklev, J. Eidson, and A. Pakdaman, "An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 5, pp. 1632–1639, Oct. 2007.

[8] P. Carbone, A. Cazzorla, P. Ferrari, A. Flammini, A. Moschitta, S. Rinaldi, *et al.*, "Low complexity UWB radios for precise wireless sensor network synchronization," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 9, pp. 2538–2548, Sep. 2013.

[9] *Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588:2008, Jul. 2008.

[10] G. Giorgi and C. Narduzzi, "Performance analysis of Kalman filter-based clock synchronization in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 8, pp. 2902–2909, Aug. 2011.

[11] C. Na, D. Obradovic, and R. L. Scheiterer, "Probabilistic model for clock synchronization of cascaded network elements," in *Proc. IMTC*, May 2009, pp. 1594–1598.

[12] D. Fontanelli and D. Macii, "Accurate time synchronization in PTP-based industrial networks with long linear paths," in *Proc. IEEE ISPCS*, Sep./Oct. 2010, pp. 97–102.

[13] D. Fontanelli, D. Macii, P. Wolfrum, D. Obradovic, and G. Steindl, "A clock state estimator for PTP time synchronization in harsh environmental conditions," in *Proc. IEEE ISPCS*, Sep. 2011, pp. 99–104.

[14] D. Fontanelli, D. Macii, S. Rinaldi, P. Ferrari, and A. Flammini, "Performance analysis of a clock state estimator for PROFINET IO IRT synchronization," in *Proc. IEEE IMTC*, May 2013, pp. 1828–1833.

[15] C. Zucca and P. Tavella, "The clock model and its relationship with the Allan and related variances," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 52, no. 2, pp. 289–296, Feb. 2005.

[16] P. Wolfrum, R. L. Scheiterer, and D. Obradovic, "An optimal control approach to clock synchronization," in *Proc. Int. IEEE Symp. Precis. Clock Synchronization Meas., Control Commun.*, Sep./Oct. 2010, pp. 122–128.

[17] D. Fontanelli, L. Greco, and L. Palopoli, "Soft real-time scheduling for embedded control systems," *Automatica*, vol. 49, pp. 2330–2338, Jul. 2013.

[18] J. Nilsson and B. Bernhardsson, "Analysis of real-time control systems with time delays," in *Proc. IEEE Conf. Decision Control*, vol. 3. Dec. 1996, pp. 3173–3178.

[19] C.-Y. Kao and A. Rantzer, "Stability analysis of systems with uncertain time-varying delays," *Automatica*, vol. 43, no. 6, pp. 959–970, Jun. 2007.

[20] M. Velasco, P. Marti, and E. Bini, "On Lyapunov sampling for event-driven controllers," in *Proc. IEEE Conf. Decision Control*, Dec. 2009, pp. 6238–6243.

[21] M. Mazo and P. Tabuada, "Input-to-state stability of self-triggered control systems," in *Proc. IEEE Conf. Decision Control*, Dec. 2009, pp. 928–933.

[22] X. Wang and M. D. Lemmon, "Decentralized event-triggered broadcasts over networked control systems," in *Proc. Int. Conf. HSCC*, Apr. 2008, pp. 674–677.

[23] A. Quagli, D. Fontanelli, L. Greco, L. Palopoli, and A. Bicchi, "Design of embedded controllers based on anytime computing," *IEEE Trans. Ind. Inf.*, vol. 6, no. 4, pp. 492–502, Nov. 2010.

[24] L. Greco, D. Fontanelli, and A. Bicchi, "Design and stability analysis for anytime control via stochastic Scheduling," *IEEE Trans. Autom. Control*, vol. 56, no. 3, pp. 571–585, Mar. 2010.

[25] J. Jasperneite, K. Shehab, and K. Weber, "Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges," in *Proc. IEEE Int. Work. Factory Commun. Syst.*, Sep. 2004, pp. 239–244.

[26] T. K. Y. Bar-Shalom, and X. Rong Li, *Estimation with Application to Tracking and Navigation–Theory, Algorithm and Software*. New York, NY, USA: Wiley, 2001.

[27] C. Na, P. Wolfrum, D. Obradovic, and R. L. Scheiterer, "Optimal estimation and control of clock synchronization following the precision time protocol," in *Proc. IEEE Multi-Conf. Syst. Control*, Sep. 2010, pp. 1767–1772.

[28] C. Na, D. Obradovic, R. L. Scheiterer, and J. A. Nossek, "A Kalman filter approach to clock synchronization of cascaded network elements," in *Proc. IFAC NecSyst. Workshop*, Sep. 2009, pp. 54–59.

[29] Y. Fang and K. Loparo, "Stochastic stability of jump linear systems," *IEEE Trans. Autom. Control*, vol. 47, no. 7, pp. 1204–1208, Jul. 2002.

[30] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802*, IEC Standard 61784-2, Jul. 2010.

[31] *IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks*, IEEE Standard 802.1Q-2005, May 2006.

[32] D. Gunzinger, C. Kuenzle, A. Schwarz, H. Doran, and K. Weber, "Optimising PROFINET IRT for fast cycle times: A proof of concept," in *Proc. IEEE Int. WFCS*, May 2010, pp. 35–42.

[33] P. Ferrari, A. Flammini, D. Marioli, S. Rinaldi, and A. Taroni, "Testing coexistence of different RTE protocols in the same network," in *Proc. IEEE Int. WFCS*, May 2008, pp. 179–187.

[34] P. Ferrari, A. Flammini, S. Rinaldi, and E. Sisinni, "On the seamless interconnection of IEEE 1588-based devices using a PROFINET IO infrastructure," *IEEE Trans. Ind. Inf.*, vol. 6, no. 3, pp. 381–392, Aug. 2010.

[35] J. C. Eidson, *Measurement, Control and Communication Using IEEE 1588*. New York, NY, USA: Springer-Verlag, 2006.

[36] K. Corell, N. Berendt, and M. Branicky, "Design considerations for software only implementations of the IEEE 1588 precision time protocol," in *Proc. Conf. IEEE–1588 Standard Precise Clock Synchron. Protocol Netw. Meas. Control Syst.*, Oct. 2005, pp. 1–6.

Daniele Fontanelli (M'09) received the M.S. degree in information engineering and the Ph.D. degree in automation, robotics and bioengineering from the University of Pisa, Pisa, Italy, in 2001 and 2006, respectively.

He was a Visiting Scientist with the Vision Laboratory, University of California at Los Angeles, Los Angeles, CA, USA, from 2006 to 2007. From 2007 to 2008, he has been an Associate Researcher with the Interdepartment Research Center E. Piaggio, University of Pisa. Since 2008, he has been an Associate Researcher with the Department of Information Engineering and Computer Science, University of Trento, Trento, Italy. His current research interests include real-time control and estimation, resource aware control, mobile robotics and visual servoing, localization algorithms, human robot interaction, and wireless sensor networks.

David Macii (M'06) received the Ph.D. degree in information engineering from the University of Perugia, Perugia, Italy, in 2003.

He was involved in research on various institutions with the Department of Digital Networks, German Aerospace Center DLR, Munich, Germany, in 2000, with the Applied DSP and VLSI Research Group, University of Westminster, London, U.K., in 2002, the Advanced Learning and Research Institute, University of Lugano, Switzerland, between 2003 and 2005, and the Berkeley Wireless Research Center, University of California at Berkeley, Berkeley, CA, USA, between 2009 and 2010, as a Fulbright Research Scholar. Currently, he is an Assistant Professor with the Department of Industrial Engineering, University of Trento. His current research interests include design, implementation, and characterization of embedded systems, with a special emphasis on power reduction and estimation techniques, distributed synchronization, and wireless sensor networks.

Stefano Rinaldi (M'11) was born in Seriate, Italy, in 1982. He received the Degree (Hons.) in electronic engineering and the Ph.D. degree in electronic instrumentation from the University of Brescia, Brescia, Italy, in 2006 and 2010, respectively.

His current research interests include performance analysis of industrial network, wireless sensor network, smart sensors, real-time Ethernet, fieldbus applications, synchronization methods, FPGA SoC design, and Linux-embedded software development.

Paolo Ferrari (S'00–M'04) was born in Italy, in 1974. He received the M.Sc. (Hons.) degree in electronic engineering and the Ph.D. degree in electronic instrumentation from the University of Brescia, Brescia, Italy, in 1999 and 2003, respectively.

He is currently an Assistant Professor with the Department of Information Engineering, University of Brescia. He is the author of more than 90 international papers. His current research interests include signal conditioning and processing for embedded measurement instrumentation, smart sensors, sensor networking, smart grids, real-time Ethernet and fieldbus applications.

Dr. Ferrari is a member of the IEC SC65C MT9, IEC TC65C WG10, and CENELEC/IEC TC65X IRWC.

Alessandra Flammini (M'99–SM'10) received the M.Sc. (Hons.) degree in physics from the University of Rome, Rome, Italy, in 1985.

She was involved with industrial research and development on digital drive control from 1985 to 1995. She is currently with the University of Brescia, Brescia, Italy, where she was an Assistant Professor from 1995 to 2002 and has been an Associate Professor since 2002. She is the responsible of the Electronics Laboratory and she has been with the National Competence Centre of PNI (Profibus Network Italia) for PROFIBUS and PROFINET since 2004. She has authored or co-authored more than 150 international papers. Her current research interests include electronic instrumentation, digital processing of sensor signals, smart sensors, and wired and wireless sensor networks with a special attention to synchronization.

Mrs. Flammini was the Conference Co-Chair of ISPCS in 2009 and SAS in 2012. Since 2013, she has been a member of the AdCom of the IEEE Instrumentation and Measurement Society.