Evaluation of EtherCAT Distributed Clock Performance

Gianluca Cena, *Senior Member, IEEE*, Ivan Cibrario Bertolotti, *Member, IEEE*, Stefano Scanzio, Adriano Valenzano, *Senior Member, IEEE*, and Claudio Zunino

***Abstract*—EtherCAT is a real-time Ethernet protocol conceived explicitly for industrial applications. It is characterized by high communication efficiency, which permits control loops to be closed with short cycle times, and is provided with a suitable mechanism, known as distributed clock (DC), that enables synchronized operations to take place across the controlled system. These features can be profitably adopted, for instance, to support motion control applications.**

In this paper, the performance of the DC mechanism is evaluated by means of a thorough campaign of experimental measurements carried out on a real network setup. A number of factors have been taken into account that can affect accuracy and precision, and their effects studied in depth.

***Index Terms*—Industrial automation, network protocols, performance and reliability.**

I. Introduction

ETHERCAT [1] is a real-time Ethernet (RTE) network that is becoming increasingly popular in factory automation environments, in particular, at the shop-floor level. Although it relies on the conventional Ethernet technology [2], the communication protocol exploits a peculiar approach to access slave devices, that resembles closely the summation frame of INTERBUS [3]. Moreover, a logical addressing scheme is defined which permits small-sized process data to be packed further. As a consequence, communication efficiency is very high (up to 90%) which achieves very short cycle times [4]. This makes this solution particularly attractive for connecting decentralized peripherals (i.e., remote I/O devices) to the application master (either a real or a soft-PLC).

Besides efficiency, another feature that makes EtherCAT appealing in a number of application domains such as, e.g., motion control, is the availability of a simple yet effective mechanism that enables devices to operate in a synchronized way (coordinate sampling and actuation).

Precise clock synchronization is a feature which is becoming more and more important in control networks. As pointed out in [5], unsynchronized networks usually suffer from non-negligible jitters. A number of approaches have been described in the literature aimed at supporting synchronized operations in popular networks such as, e.g., CAN [6], new-generation automotive networks [7], and even WLANs [8]. Many real-time Ethernet solutions rely on either the IEEE 1588 precision time protocol (PTP) [9] or its variants, such as the precision transparent clock protocol (PTCP) used in PROFINET [10], [11]. The performance of these approaches are well-known [12] and prove to be satisfactory even in s/w implementations [13]. Modifications of PTP have been defined in order to improve, e.g., fault tolerance [14] or interoperability [15].

Thanks to its ring topology, EtherCAT relies on a clock synchronization mechanism simpler than PTP, which is known as *distributed clock* (DC). In order to reduce implementation costs, supporting DC is not mandatory for EtherCAT devices. Therefore, both DC-enabled and non DC-enabled slaves are available off-the-shelf, which can quietly operate in the same network. Quite obviously, only the former kind of devices will be able to support synchronized operations.

Basically, clocks of all the DC-enabled slaves in the network are synchronized with a common timing reference under direct control of the master. The DC mechanism is able to cope with aspects such as the initial offset and drift of local oscillators, as well as the propagation delays over the network. Propagation delays and offsets are measured and compensated statically. Instead, drift correction relies on a different approach: the reference time is distributed periodically to the slaves, which tune the frequency of their local clocks consequently [16].

Despite being rather simple and straightforward, the DC mechanism enables accurate synchronization (in small-to-medium systems, clock deviations are well below 1 μs). The main goal of this paper is to assess several properties of this mechanism and, in particular, the accuracy and precision [17], [18] with which coordinate actions can be carried out by devices connected through the network. To this extent, a number of thorough measurement campaigns were carried out on real-world devices, in order to determine to which degree the actual performance matches the figures provided by data sheets.

This paper is structured as follows. In Sections II and III, the EtherCAT protocol and the DC mechanism are briefly recalled. Section IV describes our experimental setup, while Section V introduces the technique we used for analyzing DC performance. Finally, in Section VI, some differences between the behavior of Ethernet and E-bus interfaces are highlighted. Then, DC performance has been evaluated for these two kinds of interfaces separately.

II. EtherCAT Protocol Basics

The basic EtherCAT protocol only supports single-master network configurations, where the master communicates with

Manuscript received February 18, 2011; revised July 28, 2011; accepted September 04, 2011. Date of publication October 17, 2011; date of current version January 20, 2012. This work was supported in part by Regione Piemonte in the framework of the POR-FESR 2007/2013 Project Flex-Mech "New Advanced Mechanical Systems for Flexible and Customized Production" under Grant 0127000009B. Paper no. TII-11-066.

The authors are with the National Research Council of Italy, Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), I-10129 Torino, Italy (e-mail: stefano.scanzio@polito.it).

Digital Object Identifier 10.1109/TII.2011.2172434

the other devices (namely, the slaves) by sending them suitable telegrams. Each telegram encodes exactly one EtherCAT command, that is, *read*, *write* or some combination of these operations, as well as the address of the item to be accessed. In this context, an item can be either a register or a memory location on one or more devices. Two addressing modes are foreseen, that is *logical* and *physical*. The latter, in turn, can be either *configured* or *positional*.

More than one telegram can be included in the same EtherCAT frame, which yields high network throughput. In turn, each EtherCAT frame can be encapsulated in either an Ethernet frame or an UDP message. Only the first option was considered in the following, in that it ensures the highest degree of real-time performance.

EtherCAT networks are based on a physical ring topology. All EtherCAT frames are generated by the master, to which they come back after they have passed through all the slaves. Each slave processes and propagates the frame onward in a very short time (usually less than 1 μs). When an EtherCAT slave recognizes a command of its interest, it executes the related actions on-the-fly, by reading and/or changing directly parts of the Ethernet frame which is being relayed [1].

Despite, from a conceptual point of view, the topology of EtherCAT networks is an open ring closed on its ends by the master, actual network topologies look in reality a bit different. In fact, every slave is traversed by the frames twice, in opposite directions that are known as *processing* and *forwarding*, respectively. Frame processing takes place only on the former path, whereas the second is used uniquely to propagate the frames back to the master. In order to allow daisy-chain connections, every slave is provided with two network interfaces, namely, an *upstream* interface (in the direction of the master) and a *downstream* interface (in the opposite direction). This implies that, visually, the resulting network topology resembles rather a bus.

Two kinds of physical support are envisaged, that is Ethernet (both 100BASE-TX and fiber optics are allowed) and E-bus. Ethernet solutions exploit two wire pairs in the full-duplex cable to accommodate both the processing and the forwarding path in the same jacket. E-bus, on the other hand, relies on *low voltage differential signaling* (LVDS) and can be used only as a backplane for modular devices (e.g., the "terminal blocks" meant to be fastened side-by-side to a mounting rail). It is mainly intended to reduce costs, as it does not require media independent interfaces (MII) and not even transformers. Devices known as " EtherCAT couplers" (e.g., the Beckhoff EK1100) can be used to connect an Ethernet cable to a group of terminal blocks communicating over E-bus.

When a PC is used as the master device, the very first segment of the EtherCAT network relies on Ethernet. A full-duplex network interface card (NIC) is used to this purpose on the PC. Things may change when embedded devices are used such as, e.g., a PLC. In the case considered in this paper, a Beckhoff CX1020 was used that is equipped directly with an E-bus interface [Fig. 1(a)].

Hierarchical topologies can be deployed as well, by exploiting devices with more than two ports (both Ethernet and E-bus). For instance, the above described EK1100 coupler [19] is provided with two Ethernet ports (upstream and downstream) and one downstream E-bus connection. In the same way, the EK1122 ("two-port EtherCAT junction") has two E-bus connections (upstream and downstream) and two downstream Ethernet ports, whereas the EK1110 (" EtherCAT extension") has one upstream E-bus connection and one downstream Ethernet port. By exploiting these devices, line, star and tree-shaped networks can be set up [20], as shown in Fig. 1(b).

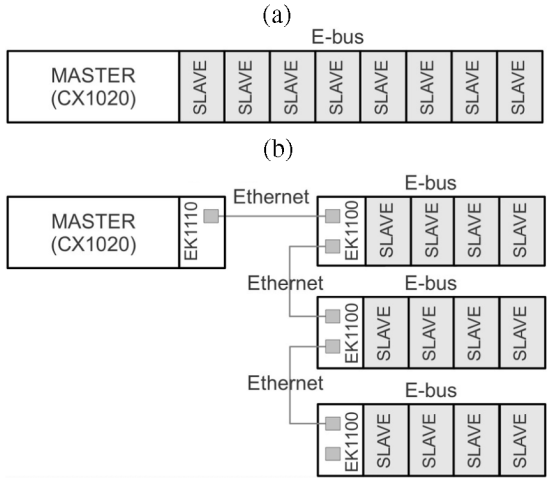

Fig. 1. EtherCAT network topologies. (a) Linear Topology. (b) Hierarchical Topology.

III. DC Mechanism Basics

EtherCAT slaves are provided with an internal clock, namely, the *local time* (t_{loc_time}), implemented as a counter that is held in the *Local_Time* register. In DC-enabled devices, a mechanism is foreseen that makes a "global clock" available to the applications, which is derived from such a counter. This global clock is denoted *system time* (t_{sys_time}) and represents the time elapsed since January 1, 2000. It is held in an internal 64-bit register, known as *System_Time*, and has a granularity of 1 ns. The system time can be used, e.g., to carry out the timestamping on sampled inputs or to actuate outputs in a coordinate way with a high degree of accuracy. Such capabilities are required in many application areas, e.g., when several axes must be moved exactly at the same time.

Actually, every EtherCAT slave is provided with a local copy of the system time, that is synchronized to the *reference time* (t_{ref_time}) over the network through the DC mechanism. The reference time coincides with the system time of the *reference clock*, which is typically the first DC-enabled slave in the network. Possible misalignments among the clocks of the slaves are due mainly to two reasons.

1) First, when slaves are switched on, the internal counters that hold the current time are reset; unfortunately, this action does not take place exactly at the same time in every slave, and this results in an initial offset among their clocks that has to be corrected.
2) The second reason is the presence of small and unavoidable differences in the slave oscillator frequencies, that make their clocks diverge over the time.

It is up to DC compensating these effects. The related algorithm consists of three phases, namely, propagation delay measurement, offset compensation, and drift compensation.

First, propagation delays are evaluated. To this extent, each slave with DC capability measures the arrival time of a specific kind of frames. This process is controlled by the master, that triggers this operation by sending broadcast write commands (BWR) addressed to a purposely defined register, i.e., the *Receive_Time_Port_0*. Whenever this telegram is received, each slave latches the time the first bit of the Ethernet preamble is read on any one of its ports in the *Receive_Time_Port* registers. For regular devices this means taking just two timestamps, on both the processing and the forwarding path. Instead, for couplers used in hierarchical network topologies, more than two time measurements are carried out. After these times have been collected from all the slaves, the master (which is aware of the network topology) computes the propagation delay for each single trunk of the bus. At the same time, it also determines the delays t_{prop_delay} measured between the *reference clock* and any slave on the forwarding path. These values are written in the *Propagation_Delay* register of every slave by means of separate write telegrams. In real networks, this phase is repeated periodically (about every 22 s).

From the above timestamps, the master also calculates the offset between the reference time and the local time of each slave (t_{offset}), and writes this value in the related *System_Time_Offset* register. This is done only once, in the setup phase, and is required in order to compensate initial offsets quickly. From now on, every slave can obtain the system time by adding in h/w the values of its *Local_Time* and *System_Time_Offset* registers

$$t_{sys\_time} = t_{loc\_time} + t_{offset}. \tag{1}$$

Once the initial offsets among the system times of the slaves have been statically corrected, the drift caused by tolerances of the internal oscillators must be compensated. This is carried out by the slaves through the *time control loop* (TCL). Since drifts depend on factors that may vary over the time like, e.g., temperature, supply voltage and ageing of components, continuous and adaptive compensation is needed throughout normal network operations. To this purpose, special EtherCAT commands known as "read multiple write" are used (either *ARMW* or *FRMW*). They are sent periodically by the master to distribute the system time of the reference clock to all the other slaves. Every slave compares the received value ($t_{recv\_sys\_time}$) to its system time (after compensating propagation delays), and determines a value Δt which is used to tune the "frequency" of its local oscillators

$$\Delta t = (t_{loc\_time} + t_{offset} - t_{prop\_delay}) - t_{recv\_sys\_time}. \tag{2}$$

The actual mechanism is indeed a bit more complex. According to the standard specification, the *Local_Time* register is updated by the TCL once every 10 ns. Its value is increased by 10 when the average value of Δt is close to 0, whereas either 11 or 9 is added when the system time is detected slower or faster than the reference clock, respectively. A filter is foreseen in the TCL to smooth abrupt changes and prevent oscillations. It takes the values of Δt as input and manages the update of the local time precisely. In order to shorten the convergence time of the DC mechanism, an initialization phase is foreseen where the master sends out a sequence of about 15 000 commands for drift compensation. The above algorithm is able to ensure that the system time of every slave increases monotonically.

A comprehensive explanation of the DC mechanism can be found in the EtherCAT standard specifications [1], as well as in the freely available data sheets of several EtherCAT slave controllers (ESC) [21]. It is worth remarking that the *System_Time* registers hold just local (well-approximated) copies of the reference time (which is the actual "global clock"). For the sake of clarity, the term "local clock" will be used in the following to denote the system time seen on any given slave and distinguish it from the global clock.

Synchronization at the application level relies on the generation of *SyncSignals* and the timestamping of *LatchSignals*. Both kinds of signals rely on the synchronized time source provided by DC (i.e., the local clock): the formers are generated internally in the ESC and are used to trigger some specific operation (actuation), whereas the latters cause the ESC to take a timestamp (sampling). In order to reduce latencies, they can be mapped directly on external signals (i.e., h/w pins) of the ESC, that is *SYNC0/1* and *LATCH0/1*, respectively [21].

IV. Experimental Setup

Measuring temporal properties concerning the Ethernet frames exchanged in a distributed system is not a trivial task [22]. The approach we followed in this paper focuses on the effects of synchronization as seen by the controlled system, i.e., on the actual sampling and actuation times in the slave devices. When dealing with DC performance, it makes little sense distinguishing between sampling and actuation. This is because, the synchronization mechanism they rely upon is exactly the same, and it is only up to the applications running in the device to decide how to exploit it.

In order to evaluate synchronization quality experimentally, two approaches can be followed:

1) configuring two distinct synchronized output devices to generate the very same nonconstant signal and measure how much their actual outputs are displaced [23];
2) feeding the same nonconstant signal to two distinct synchronized input devices and compare the timestamps they take on specific level variations [24].

In the former case, synchronized actuation is exploited, whereas the latter is based on synchronized sampling. In the following, we decided to rely on the second approach. In fact, in this case there is no need to carry out precise measurement of time offsets in the domain of "physical" signals. Instead, all calculation can be carried out on timestamps directly. As a consequence, only the errors due to the sampling accuracy of input devices may affect the experimental results. All it is required is having a generator able to produce a signal with sharp (and very steep) edges.

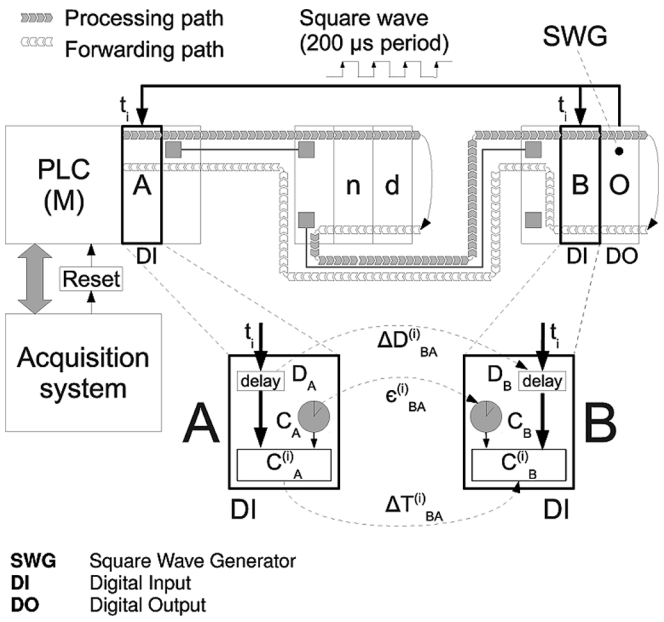

Fig. 2. Block diagram of the test-bed used for the experiments.

A. Experimental Test-Bed

A simple experimental test-bed has been developed that is made up of commonly available off-the-shelf devices, as shown in Fig. 2. The basic idea is to provide the same input (a square wave) to a pair of identical DC-enabled slaves, each of which takes a timestamp on every rising edge of this signal. In order to shorten I/O latencies, eXtreme Fast Control devices (XFC) were used, which are able to support very quick operations. Pairs of corresponding timestamps are stored in memory: information about DC performance can be obtained from the analysis of the collected values.

A square wave generator (SWG) is connected directly to two digital inputs (DI) provided with DC capability (Beckhoff EL1252 [25]) through wires having the same length. DIs are managed through an EtherCAT master (M), that is a Beckhoff CX1020 embedded PC [26] provided with E-bus interface. SWG has been implemented using EtherCAT technology. A 5 kHz square wave is generated by sending one frame every 100 μs to a nonsynchronized digital output (DO) (Beckhoff EL2202 [27]), to make its output channel switch alternatively on and off. It is worth noting that the accuracy of the period of the square wave does not affect in any way the results of the following experiments. For this reason, the same master M used to collect timestamps from DIs has been used to drive DO as well. For the sake of clarity, the DIs used to collect timestamps will be denoted in the following A and B, respectively, whereas the DO will be denoted O. Other slaves added to this basic configuration will be referred to as either d (DC-enabled), or n/m (non-DC-enabled).

Each DI uses its internal synchronized local clock (system time) for taking timestamps. A PLC program, which runs on M, was purposely developed in order to acquire the timestamps taken by A and B and collect suitable statistics on them (preprocessing is required because the main memory of the PLC has reduced storage capabilities). When the experiment ends, data are transferred automatically to a PC and then postprocessed offline to obtain the required performance indices.

Since O is connected directly to A and B through a pair of identical wires, the timestamps they take on the same edge should be, in theory, identical. Any difference between them reflects the inaccuracy of distributed synchronization in EtherCAT and depends basically on two causes:

1) the unavoidable h/w differences between A and B, which implies that their input channel circuitries may introduce slightly different delays on edge detection;
2) the actual capabilities of the DC mechanism, which is the aspect we are mostly interested in.

Since the $Local\_Time$ register in the slaves is updated every 10 ns, all time measurements are affected by a quantization error. However, because of the way the DC mechanism operates, such an error is not independent at all of the synchronization error. This means, that their effects can hardly be studied separately. Therefore, any evaluation of DC performance carried out by considering the slaves as "black boxes" necessarily accounts for both these sources of uncertainty at the same time.

B. Measurement Technique

What we are interested in here is evaluating if and how much the network topology affects DC accuracy and precision. In particular, the effects of the order, number and position of slaves on the bus have been taken into account, as well as their interface (Ethernet versus E-bus). In order to describe quickly EtherCAT network topologies, devices are specified in the same order as they appear on the bus (and are traversed by the frames). The dash " – " is used to indicate an Ethernet cable. For instance, "*MA-BO*" means: the PLC M, connected to DI A on E-bus, then an Ethernet cable (preceded by an EtherCAT extension and followed by an EtherCAT coupler); then, on E-bus again, DI B and, finally, the DO O.

The basic experiment consisted in evaluating the difference between the timestamps taken by the two DC-synchronized devices A and B on the same external input. In every experiment, 4 million measurements were taken in sequence, and suitable statistical indices evaluated on the set of the acquired samples. In order to let the DC mechanism settle, every test was preceded by a warm-up phase (about 5-min long), where about 1.5 million samples were acquired and discarded. Such a time proved to be sufficient so that, in almost every case, any transient behavior due to the startup of the DC mechanism disappears before the actual measurement begins.

In order to characterize the outcomes of the experiments, the average value μ, standard deviation σ, maximum $\max$, and minimum $\min$ of the above differences, as well as their statistical distribution (i.e., probability density function, pdf) were chosen. It is worth pointing out that the evaluation of $\max$ and $\min$ is essential in order to assess whether or not the DC mechanism is able to meet real-time requirements.

V. Analysis of the Measured Values

In the following, unless otherwise stated, all the slaves are assumed to support the DC mechanism. Let C_X be the clock as seen by slave X. In particular, $C_A(t)$ and $C_B(t)$ are the values of the local clocks of A and B at time t, as shown in Fig. 2.

Since the DC mechanism is not perfect, likely they will not be perfectly aligned. What we are going to evaluate in the following is the synchronization error ϵ. At any time t and for any pair of slaves A and B, it is defined as

$$\epsilon_{BA}(t) \triangleq C_B(t) - C_A(t). \tag{3}$$

The square wave fed into A and B generates an ordered sequence of events, each one corresponding to a single rising edge. Let t_i be the (absolute) time when the ith event takes place. Each one of such events causes both A and B to take a timestamp, indicated as $C_A^{(i)}$ and $C_B^{(i)}$, respectively. These values are not the same as $C_A(t_i)$ and $C_B(t_i)$, because of the latencies introduced by the input channels of DIs.

Generally speaking, in all of the following experiments, the quantity actually measured is $\Delta T_{YX}^{(i)}$, defined as the difference between the timestamps in the pair $\left\langle C_Y^{(i)}, C_X^{(i)} \right\rangle$

$$\Delta T_{YX}^{(i)} \triangleq C_Y^{(i)} - C_X^{(i)} \tag{4}$$

where X is the slave closer to the master, while Y is the farther one. The specific network configuration is not indicated explicitly in order to keep the notation as simple as possible.

Because of the sampling delays introduced by the input circuitry, timestamping on DIs does not take place at exactly the same time. Instead, it is actually triggered on slave X at time $t_i + D_X$, where D_X is the latency of the related input channel. The first experiment is aimed at evaluating the discrepancies between such latencies. To this extent, two simple network configurations were considered (*MABO* and *MBAO*). In the first case, the reference clock is A, whereas in the second it is B. Reasonably, the input latencies D_A and D_B can be assumed not to vary over the time. Let ΔD_{BA} represent their difference

$$\Delta D_{BA} \triangleq D_B - D_A. \tag{5}$$

Usually, the quality of local oscillators is very good (some tens p.p.m.), input latencies very short (below 1 μs for fast devices) and local clocks are adjusted by the DC control algorithm smoothly (because of the TCL). Therefore, the drift of any local clock C_X over a time interval D_X can be quietly neglected, which implies

$$C_X^{(i)} = C_X(t_i + D_X) \cong C_X(t_i) + D_X. \tag{6}$$

As a consequence, from now on it will be assumed that

$$\Delta T_{YX}^{(i)} = (C_Y(t_i) + D_Y) - (C_X(t_i) + D_X). \tag{7}$$

Let $\epsilon_{YX}^{(i)}$ be the deviation of the local clock of Y with respect to X (that, in this case, is the reference clock) at time t_i

$$\epsilon_{YX}^{(i)} \triangleq \epsilon_{YX}(t_i) = C_Y(t_i) - C_X(t_i). \tag{8}$$

Then

$$\Delta T_{YX}^{(i)} = \epsilon_{YX}^{(i)} + \Delta D_{YX}. \tag{9}$$

A large number of data points were acquired, by considering both configurations (*MABO* and *MBAO*) and all the possible combinations of input channels (each DI has two distinct ones).

TABLE I
STATISTICAL PROPERTIES OF ΔT WHEN CHANGING DI CHANNELS AND THE ORDER OF A AND B ON THE BUS

Configuration	Channel	$E[\Delta T]$ (ns)	$\sigma_{\Delta T}$ (ns)	Results (ns)
MABO (ΔT_{BA})	A = ch. 1	8.34	4.25	$\overline{\epsilon} = -4.58$
MBAO ($\Delta T'_{AB}$)	B = ch. 1	-17.53	4.27	$\Delta D_{B_1A_1} = 12.94$
MABO (ΔT_{BA})	A = ch. 2	25.41	4.33	$\overline{\epsilon} = -3.10$
MBAO ($\Delta T'_{AB}$)	B = ch. 2	-31.60	4.39	$\Delta D_{B_2A_2} = 28.51$
MABO (ΔT_{BA})	A = ch. 2	16.85	4.29	$\overline{\epsilon} = -3.31$
MBAO ($\Delta T'_{AB}$)	B = ch. 1	-23.47	4.34	$\Delta D_{B_1A_2} = 20.16$
MABO (ΔT_{BA})	A = ch. 1	14.23	4.30	$\overline{\epsilon} = -3.59$
MBAO ($\Delta T'_{AB}$)	B = ch. 2	-21.42	4.39	$\Delta D_{B_2A_1} = 17.83$

Overall, this produces eight data sets, as shown in Table I. For each data set, the average value and standard deviation of ΔT have been calculated. From (9), in the *MABO* case

$$E[\Delta T_{BA}] = E[\epsilon_{BA}] + \Delta D_{BA} \tag{10}$$

whereas for the second network configuration *MBAO*

$$E[\Delta T'_{AB}] = E[\epsilon'_{AB}] - \Delta D_{BA}. \tag{11}$$

It is worth noting that the values $\Delta T_{BA}^{(i)}$ and $\Delta T_{AB}^{(i)\prime}$ actually refer to two different network configurations, so their expected values will usually differ. Instead, $E[\epsilon'_{AB}]$ in the *MBAO* case is reasonably the same as $E[\epsilon_{BA}]$ in the *MABO* case. In fact, they depend only on the DC mechanism (which is implemented exactly in the same way in both A and B) and the two network configurations were simply obtained by swapping the position of A and B over the bus. For clarity, the symbol $\overline{\epsilon}$ is used to refer to both quantities

$$\overline{\epsilon} = E[\epsilon'_{AB}] = E[\epsilon_{BA}]. \tag{12}$$

It is possible to evaluate both $\overline{\epsilon}$ and ΔD_{BA} by solving the following linear system:

$$\begin{bmatrix} E[\Delta T_{BA}] \\ E[\Delta T'_{AB}] \end{bmatrix} = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \begin{bmatrix} \overline{\epsilon} \\ \Delta D_{BA} \end{bmatrix}. \tag{13}$$

By repeating the same process for the four combinations of channels 1 and 2 in both the slaves, the results shown in the rightmost column of Table I are obtained. It is clear that:

1) The accuracy $\overline{\epsilon}$ of the distributed clock in the specific case of two adjacent slaves on E-bus is very good (the downstream slave is about 4 ns late).
2) The input channel latencies of the devices we used in our experiments differ by few tens ns; it is worth noting that, in previous tests we carried out with other devices, this difference was as high as about 100 ns [24].

Unfortunately, with this experiment it is not possible to evaluate sampling delays for the different channels separately. However, the values computed for ΔD_{BA} can be used to compensate experimental measurements concerning mean time differences. In particular, $\overline{\epsilon}$ can be obtained from $E[\Delta T]$ through (13). Conversely, σ_ϵ is exactly the same as $\sigma_{\Delta T}$.

From now on, slave A precedes slave B on the bus in every experiment. Therefore, the difference between the local clocks of B and A at time t_i will be denoted simply as $\epsilon^{(i)}$. Moreover,

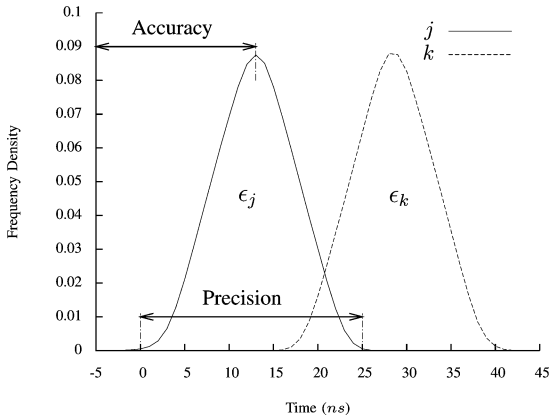

Fig. 3. Distribution of synchronization error ϵ for two different instances (j and k) of the same experiment when using Ethernet.

channels labeled as "1" were always used on both A and B. As a consequence, the value for ΔD_{BA} in the first row of the table (i.e., 13 ns) has been used for compensation.

VI. Experimental Results

In order to characterize the repeatability of measurements, each network configuration has to be tested more than once. Let n be the number of times the same experiment is repeated within any one of such "measurement campaigns." The outcome of the jth experiment in a campaign is characterized by a tuple of statistical indexes ($\overline{\epsilon}_j$, σ_{ϵ_j}, $\min_{\epsilon_j}$ and $\max_{\epsilon_j}$) or, alternatively, by the distribution of the error ϵ_j. As can be seen in Fig. 3, the shape of the distributions we found when repeating the same experiment is almost the same (basically, a Gaussian), as well as their statistical "width" σ_{ϵ_j} (*DC precision* [18]). On the other hand, mean values $\overline{\epsilon}_j$ (*DC accuracy* [18]) may, quite surprisingly, change in configurations where Ethernet interfaces were involved. For example, Fig. 3 shows the distribution of ϵ when the same experiment (*MA-BO*) is repeated twice. In order to understand how Ethernet couplers worsen DC accuracy, measurement campaigns were carried out for some basic configurations.

Because of the high number of samples acquired in each experiment (4 million) and the fact their standard deviation σ_{ϵ_j} is quite low (below 5 ns), each value $\overline{\epsilon}_j$ can be considered, for the purpose of this analysis, a reliable estimate of the accuracy achieved by DC specifically in that experiment. For the sake of clarity, δ_j is used to denote the average synchronization error in experiment j

$$\delta_j = \overline{\epsilon}_j. \tag{14}$$

Let us consider δ_j as the basic sample for a new kind of experiment (i.e., the measurement campaign). Then, $\delta_1, \delta_2, \ldots, \delta_n$ can be seen as a set of n samples pertaining to the campaign. Since the time needed to obtain each one of such samples is quite long (about half an hour), experiments in a campaign can be realistically repeated only a limited number of times (few hundreds at most). For the above set, the expected value $E[\delta]$ and standard deviation σ_δ can be calculated in turn.

In order to assess the reliability of the estimated value $E[\delta]$, confidence intervals [28] were evaluated as

$$E[\delta] \pm t_{n-1,\alpha/2} \cdot \sqrt{\frac{\sigma_\delta^2}{n}} \tag{15}$$

TABLE II
Statistical Properties of the Average Synchronization Error With Different Configurations Based on E-Bus and Ethernet

Parameters		Configuration		
		MABO	*MA-BO*	*MA- - -BO*
n		100	800	200
$E[\delta]$	(ns)	-4.60	19.87	61.57
$CI_{95\%}$	(ns)	$\pm$ 0.05	$\pm$ 0.34	$\pm$ 1.17
σ_δ	(ns)	0.27	5.81	9.98
ϵ_{min}	(ns)	-20	-13	18
ϵ_{max}	(ns)	11	51	105
$max_{\Delta\epsilon}$	(ns)	30	46	38
$\Delta\delta$	(ns)	1.39	20.51	50.94

where $t_{n-1,\alpha/2}$ is the quantile of order $1-\alpha$ of a Student's t-distribution with $n-1$ degrees of freedom. In our analysis, 95% confidence intervals were chosen ($CI_{95\%}$). This means, the probability that the actual expected value for δ is included in the interval given in (15) is 95%.

In order to characterize the real-time behavior of DC, worst-case parameters have to be taken into account.

- $\epsilon_{\max} = \max_{j=1,\ldots,n}(\max_{\epsilon_j})$ is the maximum error when considering all the experiments in a campaign, while $\epsilon_{\min} = \min_{j=1,\ldots,n}(\min_{\epsilon_j})$ is the minimum.
- $\Delta\epsilon_j = \max_{\epsilon_j} - \min_{\epsilon_j}$ is the width of the interval related to experiment j where all $\epsilon_j^{(i)}$ values lie, while $\max_{\Delta\epsilon} = \max_{j=1,\ldots,n}(\Delta\epsilon_j)$ is the largest among all intervals $\Delta\epsilon_j$.
- $\Delta\delta = \max_{j=1,\ldots,n}(\delta_j) - \min_{j=1,\ldots,n}(\delta_j)$ is the width of the distribution δ.

A. Difference Between E-Bus and Ethernet

Table II shows the results obtained for three different network configurations, that is *MABO*, *MA-BO*, and *MA---BO*. The former relies only on E-bus [Fig. 1(a)], while the latter two rely on Ethernet; in the second case only one coupler is foreseen, whereas a line structure with three couplers is envisaged in the third case [Fig. 1(b)].

The results we found when repeating several times (~ 100) the first experiment (*MABO*) were always very similar, as shown in the leftmost column in Table II. The standard deviation ($\sigma_\delta = 0.27$ ns) is very small: this means, that for our purposes a single experiment is likely sufficient to characterize E-bus behaviour satisfactorily. Thus, all the campaigns on E-bus reported in Section VI-B include only one experiment.

Two kinds of campaigns were considered for Ethernet.

1) *warm-reset*: the PLC (i.e., the EtherCAT master) is logically reset between subsequent experiments by invoking the *NT_Reboot* function block available in the *TcUtilities.lib* library of TwinCAT.
2) *cold-reset*: the PLC is physically reset through a relay controlled by the PC that collects measurements; the entire EtherCAT system is powered down and kept off for 5 min between subsequent experiments.

Warm-reset results are not reported here, since no unexpected behavior was found. Instead, results in the cold-reset case are summarized in Table II. Experiments were repeated many times in order to obtain a meaningful set of samples.

DC accuracy worsens in this case: in fact, the expected value $E[\delta]$ of the mean difference between clocks is, overall, 19.87 ns for *MA-BO* and 61.57 ns for *MA---BO*. Fig. 4 depicts the *pdf*

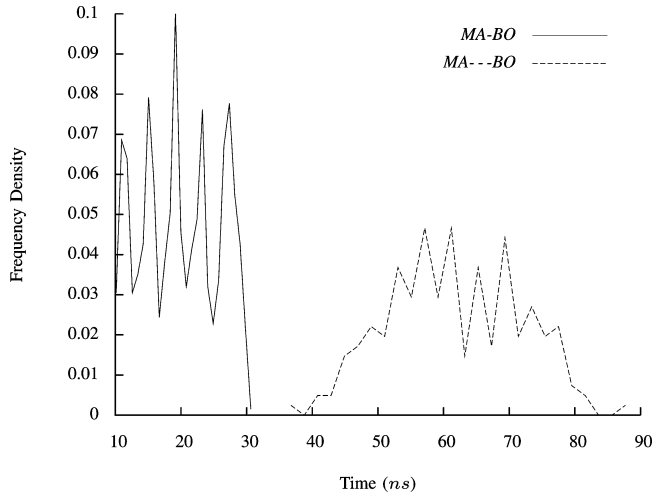

Fig. 4. Distribution of average synchronization error δ with different configurations that use Ethernet.

of the mean synchronization error δ for the two campaigns that concerned Ethernet. The width $\Delta\delta$ of the distribution is 20.51 ns for *MA-BO* and 50.94 ns for *MA---BO*, as reported in the last row of Table II. This is noticeably larger than E-bus (i.e., 1.39 ns for *MABO*). By taking confidence intervals into account and because of the non-negligible standard deviation of δ, we decided that every experiment on Ethernet should be repeated (at least) 30 times so as to obtain meaningful results. This rule was obeyed for all the configurations considered in Section VI-C.

Concerning the real-time behavior, all the steady-state synchronization errors over Ethernet lay in the range $[-13\ \text{ns}, +51\ \text{ns}]$ for *MA-BO* and $[+18\ \text{ns}, +105\ \text{ns}]$ for *MA---BO*. Despite this is worse than E-bus (for *MABO* all errors $\epsilon^{(i)}$ fell into the range $[-20\ \text{ns}, +11\ \text{ns}]$), differences among clocks remained always confined to a narrow interval. Since these results were obtained over a long period of time (about 30 h for *MABO* and 250 h for *MA-BO*), they are quite reliable. Concerning any single experiment, the width of error distributions for Ethernet is a bit larger than for E-bus (see $\max_{\Delta\epsilon}$ in the last but one row of Table II). This confirms that, generally speaking, DC worsens over Ethernet.

A formal proof of this behavior can hardly be provided, as we do not know the internal structure of the slaves exactly. A possible explanation about the reasons why the same experiment may converge to different average values δ_j for slaves with Ethernet interfaces is as follows. When a frame enters a slave on the processing path, it passes through the upstream MII, the EtherCAT Processing Unit (EPU) and, finally, the downstream MII. A buffer is foreseen between the first MII and the EPU, so as to decouple them (the sender and the receiver of the frame are based on different clock sources). Conversely, data are sent from the EPU to the second MII directly (4 bits at a time) for performance reasons. Both the EPU and MIIs share the same clock source at 25 MHz, which is used, inside these devices, to obtain higher clock frequencies. Hence, the clocks used by the EPU and MIIs may suffer from a phase offset (up to 4 bit times) that could differ every time the slave is switched on (but remains the same throughout normal slave operation). The same happens to the pass-through delays: this leads, potentially, to an asymmetry between the propagation delays in the two directions that may differ after every cold reset (or when the communication link is temporarily lost), therefore influencing accuracy. Despite it is not compensated by the basic DC mechanism, it does not affect precision (see Fig. 3).

TABLE III
STATISTICAL PROPERTIES OF SYNCHRONIZATION ERROR OVER E-BUS

Set	Configuration	$\overline{\epsilon}$	σ_ϵ	min_ϵ	max_ϵ
		(ns)			
1	***MABO***	**-4.66**	**4.25**	**11**	***-18***
	MAnnnnnBO	-20.42	4.29	-5	-35
	MAdddddBO	-18.37	4.33	-3	-34
	MAmmmmBO	16.83	4.30	31	1
	MAnnnnnddddBO	-34.04	4.43	-17	-52
	MAdddddnnnnnBO	-35.78	4.44	-20	-53
	MAnnnnnddddmmmmBO	-14.56	4.60	5	-31
2	***MABO***	**-4.66**	**4.25**	**11**	**-18**
	MABOnnnnn	-5.20	4.24	9	-19
	MABOnnnnnddddd	-4.80	4.21	9	-18
	MABOnnnnnddddmmmm	-4.11	4.24	10	-18
3	*MABOnnnnnmmmm*	-5.22	4.24	8	-20
	MnnnABOnnmmmm	-7.98	4.22	6	-21
	MnnnnnmABOmmm	-7.69	4.21	6	-21
	MnnnnnmmmmABO	-4.77	4.25	9	-18

B. E-Bus

A number of experiments were carried out when devices are connected over E-bus. Results are shows in Table III. When describing network configurations, *ddddd* represents a sequence of 5 DC-enabled slaves, whereas *nnnnn* and *mmmm* refer to two sequences of devices with no DC support, made up of 5 and 4 slaves, respectively.

The first kind of experiments (set 1 in Table III) was aimed at determining how much the DC mechanism is affected by the "distance" from the reference clock (i.e., the number of slaves located between A and B). Each block of slaves seems to worsen accuracy, by displacing $\overline{\epsilon}$ (as well as the minimum and maximum) by a specific offset. As can be seen from results, the contribution of each block is, more or less, always the same (-15.76 ns for *nnnnn*, -13.71 ns for *ddddd* and $+21.49$ ns for *mmmm*), even though its exact value can hardly be explained. This means, that an "additive" property holds for DC accuracy when composing modular devices over E-bus. Moreover, results do not change when the order of blocks is swapped (rows 5 and 6). As a consequence, a "commutative" property seems to hold as well. Standard deviation σ_ϵ only increases slightly when additional slaves are added. This means, that contributions are not statistically independent, and the DC mechanism actually manages to preserve precision.

The second experiment (set 2 in Table III) dealt with the overall number of slaves when the relative position of A and B is left unchanged. In particular, a variable number of devices were connected after B. As can be seen, average, maximum and minimum values, as well as standard deviation only change slightly. This means, that this parameter does not have appreciable effects on DC accuracy and precision.

The last experiment (set 3 in Table III) was aimed at measuring the effect of the position on the bus of A and B when their relative distance does not change. A remains in every case the reference clock, since the slaves that are connected before it (both n and m) are not DC enabled. Again, it seems that DC is not affected noticeably.

TABLE IV
STATISTICAL PROPERTIES OF AVERAGE SYNCHRONIZATION ERROR OVER ETHERNET (EXPERIMENTS ARE REPEATED SEVERAL TIMES)

Set	Configuration	n	$E[\delta]$	$CI_{95\%}$	σ_δ	ϵ_{min} (ns)	ϵ_{max}	$max_{\Delta\epsilon}$	$\Delta\delta$
1	*M-ABO*	30	-7.41	± 0.09	0.29	-21	8	28	1.29
	MABO-	30	-5.70	± 0.08	0.24	-20	10	29	1.17
	Md-ABO	30	-6.23	± 0.07	0.22	-22	10	31	1.00
2	***MA-BO*** (*C1*)	**30**	**19.49**	**± 1.70**	**5.29**	**-11**	**47**	**41**	**16.91**
	MA-BO (*C2*)	30	20.86	± 1.70	5.31	-11	51	49	16.50
	MA-BO (*C3*)	30	-8.06	± 0.09	0.29	-29	10	37	1.21
3	***MA-BO*** (*C1*)	**30**	**19.49**	**± 1.70**	**5.29**	**-11**	**47**	**41**	**16.91**
	MA--BO (*C3 C1*)	30	40.11	± 1.71	5.33	7	66	46	15.94
	MA---BO (*C3 C2 C1*)	30	63.94	± 3.44	10.75	23	97	36	40.47
4	***MA---BO***	**30**	**63.94**	**± 3.44**	**10.75**	**23**	**97**	**36**	**40.47**
	MA---mmmmBO	30	73.59	± 2.77	8.64	43	112	36	37.87
	MA---nnnnnddddmmmmBO	30	36.67	± 3.80	11.84	-3	78	38	48.22
5	*MA-BO (C1, Temporary link fault)*	30	19.73	± 1.63	5.25	-6	43	41	16.86

C. Ethernet

Here, DC performance is evaluated when devices are connected through Ethernet interfaces. As said before, every experiment concerning a specific network configuration has been repeated several times (30) in order to obtain significant results, and the whole system was powered down and up before each single test. Results are shown in Table IV, together with 95% confidence intervals. The configuration $MA\text{-}BO$, where the Ethernet coupler is placed between A and B, was taken as the reference for the experiments in sets 1, 2, and 3.

A first set of experiments (set 1 in Table IV) was aimed to analyse the effects on DC of the position of the coupler with respect to A and B. Configurations *M-ABO* and *MABO-*, where the coupler is located before and after the AB pair, respectively, showed results very close to E-bus (*MABO*). This is not surprising since, as explained before, the coupler has no effect in that it is not placed between A and B. The same also happened in the *Md-ABO* case, where the reference clock is the slave d and is separated by the AB pair through a coupler. Here, both A and B suffer from the same random offset added by the coupler each time it is powered on. Hence, their accuracy is affected in the same way. Other statistical indices are comparable with the *MABO* case; so, the insertion of a coupler in these configurations does not affect DC.

The second kind of experiments (set 2 in Table IV) analyzed the differences among couplers. The coupler $C1$ in the *MA-BO* configuration was replaced by two other couplers of the same type ($C2$ and $C3$). While $C2$ behaved essentially the same as $C1$, the behavior of $C3$ was quite different. In fact, the contribution of both $C1$ and $C2$ to the average error δ (when compared to the *MABO* case) is $\sim 24\,\mathrm{ns}$, whereas for $C3$ it is about $-4.5\,\mathrm{ns}$. Another difference between these couplers concerns their behavior when the same experiment is repeated several times, as can be seen from σ_δ and $\Delta(\delta)$ values. It seems $C3$ is more deterministic (i.e., it features better repeatability of outcomes). Even if both the EEPROM content and revision number (*EK1100-0000-0017*) are the same for all couplers, $C1$ and $C2$ belong to a different batch than $C3$. Therefore, the latter (which is the most recent one) might actually be not exactly the same as the former two. A possible reason about the above difference is that, $C3$ could perform in a more effective way the compensation of delays between EPU and MII (see the end of Section VI-A).

In the third set of experiments (set 3 in Table IV), the number of couplers located between A and B was varied. The values of $E[\delta]$ show that, each coupler worsens mean accuracy by $\sim 23\,\mathrm{ns}$. It should be noted, that in the tests of set 2 $C3$ just added about $-4.5\,\mathrm{ns}$. A reasonable explanation of this behavior is that, in that experiment, only the upstream MII of the coupler and the E-bus connection (i.e., port 0 and port 1 of the ESC) were used, while in this configuration both its MIIs (i.e., port 0 and port 2) are used, while E-bus is disconnected. Analyzing σ_δ and $\Delta\delta$ and keeping in mind the peculiarities of $C3$, it can be seen that $C1$ and $C2$ worsen the precision of DC, while $C3$ leaves it practically unchanged (see the second and third case of set 3). As can be seen in the last experiment of the set, where both $E[\delta]$ and σ_δ increase, when the number of couplers grows both accuracy and precision worsen. In case this trend is confirmed when additional couplers are connected (this "only" requires to buy many other couplers), this would mean that the synchronization error (accuracy and, for older devices, precision as well) depends directly on the number of couplers. For the sake of truth, it has to be remarked that it grows quite slowly.

In set 4, the combined effect of Ethernet and E-bus interfaces is considered; results somehow confirms the above conjecture.

D. Other Factors

Finally, a number of experiments were performed to draw out results when other factors are taken into account that affect the network environment, such as, e.g., temporary faults on network links. Losing some frame from time to time (a rare but unavoidable event in real networks) does not affect synchronization appreciably. On the other hand, when the EtherCAT link remains down for some time, the free running counters in the slaves are no longer synchronized and tend to diverge (and their local clocks as well). As soon as the link is reenabled (which also implies Ethernet autonegotiation), the master enforces a procedure in order to restart the DC mechanism properly. Our aim is to assess to which extent accuracy and precision are affected by such events.

The experiment is basically the same as the $MA\text{-}BO$ configuration described above and was repeated 30 times. Between any two consecutive experiments, the Ethernet cable was unplugged for about 10 min and then plugged in again. Such a time is sufficient so that timeouts expire in both the master and

all the slaves. Results (set 5 in Table IV) show that the unavailability of the communication link leads to effects similar to those seen in the case when the system was physically reset. In particular, accuracy worsens in these experiments, which means that such temporary errors may affect the synchronization quality in steady-state conditions.

VII. Conclusion

In this paper, the performance of the synchronization technique of EtherCAT, namely the Distributed Clock, is evaluated and the influence of network topology and interface types on its accuracy, precision and real-time behavior analyzed. To this purpose, very long measurement campaigns were carried out, that involved tests on real network setups that lasted for hundreds of hours and involved (overall) billions of samples. As a consequence, we are confident that the results we obtained are statistically significant.

The quantity we evaluated is basically the synchronization error (i.e., the offset among local clocks). While average values and standard deviation provide a useful estimate about the accuracy and precision, respectively, maximum and minimum values are required to assess whether or not the DC mechanism is able to meet hard real-time constraints. Despite experimentation was necessarily carried out on small test-beds (about 20 devices at most), results show that DC performance on E-bus is actually very good. Things worsen a bit when Ethernet interfaces are involved, which also showed an unexpected (but not critical) behavior. All in all, DC seems to be able to support applications that require distributed synchronization (e.g., motion control) in a proper way.

References

[1] "Industrial Communication Networks—Fieldbus Specifications—Part 3–12: Data-Link Layer Service Definition—Part 4–12: Data-Link Layer Protocol Specification—Type 12 Elements," International Electrotechnical Commission (IEC), 61158-3/4-12 (Ed. 1.0), 2007.

[2] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements, Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Standard 802.3-2008 (Revision of IEEE Standard 802.3-2005), 2008.

[3] "Industrial Communication Networks—Fieldbus—Communication Profile Family 6: INTERBUS," International Electrotechnical Commission (IEC), FIELDB-CPF-6 (Ed. 1.0), 2008.

[4] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *Proc. 13th IEEE Conf. Emerging Technol. Factory Autom., ETFA'08*, Sep. 2008, pp. 408–415.

[5] P. Ferrari, A. Flammini, D. Marioli, A. Taroni, and F. Venturini, "Experimental analysis to estimate jitter in PROFINET IO class 1 networks," in *Proc. 11th IEEE Int. Conf. Emerging Technol. Factory Autom., ETFA'06*, Sep. 2006, pp. 429–432.

[6] G. Rodriguez-Navas, S. Roca, and J. Proenza, "Orthogonal, fault-tolerant, and high-precision clock synchronization for the controller area network," *IEEE Trans. Ind. Informat.*, vol. 4, no. 2, pp. 92–101, May 2008.

[7] M. Fugger, E. Armengaud, and A. Steininger, "Safely stimulating the clock synchronization algorithm in time-triggered systems—A combined formal and experimental approach," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 132–146, May 2009.

[8] A. Mahmood, G. Gaderer, and P. Loschmidt, "Clock synchronization in wireless LANs without hardware support," in *Proc. IEEE Int. Workshop on Factory Commun. Syst., WFCS'08*, May 2010, pp. 75–78.

[9] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," vol. 1, pp. 1–269, 2008.

[10] "Industrial Communication Networks—Profiles Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3," International Electrotechnical Commission (IEC), IEC 61784-2 (Ed. 1), 2007.

[11] "Industrial Communication Networks—Fieldbus Specification," International Electrotechnical Commission (IEC), IEC 61158 family (Ed. 2), 2007.

[12] R. Exel and G. Gaderer, "Boundaries of Ethernet layer 2 hardware timestamping," in *Proc. IEEE Int. Workshop on Factory Commun. Syst., WFCS'08*, May 2008, pp. 255–258.

[13] G. Cena, M. Cereia, I. Cibrario Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "A software implementation of IEEE 1588 on RTAI/RTnet platforms," in *Proc. 15th IEEE Int. Conf. Emerging Technol. Factory Autom., ETFA'10*, Sep. 2010, pp. 1–8.

[14] G. Gaderer, P. Loschmidt, and T. Sauter, "Improving fault tolerance in high-precision clock synchronization," *IEEE Trans. Ind. Informat.*, vol. 6, no. 2, pp. 206–215, May 2010.

[15] P. Ferrari, A. Flammini, S. Rinaldi, and E. Sisinni, "On the seamless interconnection of IEEE1588-based devices using a PROFINET IO infrastructure," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 381–392, Aug. 2010.

[16] G. Prytz and J. Skaalvik, "Redundant and synchronized EtherCAT network," in *Proc. Int. Symp. Ind. Embedded Syst., SIES'10*, Jul. 2010, pp. 201–204.

[17] J. R. Taylor, *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. Mill Valley, CA: Univ. Science Books, 1996.

[18] B. N. Taylor and C. E. Kuyatt, "Guidelines for evaluating and expressing the uncertainty of NIST measurement results." 1994. [Online]. Available: http://www.nist.gov/pml/pubs/tn1297/index.cfm

[19] "EK1100 EtherCAT bus coupler," Beckhoff Automation GmbH, 2009 [Online]. Available: http://www.beckhoff.com/

[20] M. Knezic, B. Dokic, and Z. Ivanovic, "Topology aspects in EtherCAT networks," in *Proc. 14th Intl. Power Electron. Motion Control Conf., EPE/PEMC'10*, Sep. 2010, pp. T1-1–T1-6.

[21] "Hardware data sheet ET1100—EtherCAT slave controller, Ver. 1.8," Beckhoff Automation GmbH, 2009 [Online]. Available: http://www.beckhoff.com/

[22] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "A distributed instrument for performance analysis of real-time Ethernet networks," *IEEE Trans. Ind. Informat.*, vol. 4, no. 1, pp. 16–25, Feb. 2008.

[23] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Performance evaluation of the EtherCAT distributed clock algorithm," in *Proc. IEEE Int. Symp. Ind. Electron., ISIE'10*, Jul. 2010, pp. 3398–3403.

[24] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "On the accuracy of the distributed clock mechanism in EtherCAT," in *Proc. IEEE Int. Workshop on Factory Commun. Syst., WFCS'10*, May 2010, pp. 43–52.

[25] "EL1252 2-Channel digital input terminal with time stamp," Beckhoff Automation GmbH, 2011. [Online]. Available: http://www.beckhoff.com/

[26] "CX1020 Basic CPU Module," Beckhoff Automation GmbH. [Online]. Available: http://www.beckhoff.com/

[27] "EL2202 2-Channel Digital Output Terminal," Beckhoff Automation GmbH, 2011. [Online]. Available: http://www.beckhoff.com/

[28] M. Smithson, *Confidence Intervals*, ser. Sage University Papers Series on Quantitative Applications in the Social Sciences. Houston, TX: Sage, 2003.

Gianluca Cena (SM'09) received the Laurea degree in electronic engineering and the Ph.D. degree in information and system engineering from the Politecnico di Torino, Torino, Italy, in 1991 and 1996, respectively.

In 1995, he became Assistant Professor at the Department of Computer Engineering, Politecnico di Torino and, in 2001, he joined the National Research Council of Italy (CNR) as a Senior Researcher. Since 2005, he has been Director of Research with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (CNR- IEIIT), where he is engaged in research activities concerning communications in automated factory environments. He coauthored about 100 technical papers in the area of computer communications. His current research interests include industrial communication systems, wired and wireless networks, and real-time protocols. In the past decade, he has been teaching courses on industrial communications at the Third School of Engineering—Information Technologies of the Politecnico di Torino.

Prof. Cena served as Program Co-Chairman for the 2006 and 2008 editions of the IEEE Workshop on Factory Communication Systems. He has been Associate Editor of the IEEE Transactions on Industrial Informatics since 2009.

Ivan Cibrario Bertolotti (M'06) received the Laurea degree (*summa cum laude*) in computer science from the University of Torino, Torino, Italy, in 1996.

Since then, he has been a Researcher with the National Research Council of Italy (CNR). Currently, he is with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), Torino. He teaches several courses on real-time operating systems at Politecnico di Torino, Torino, and has served as a technical referee for several international conferences and journals. His current research interests include real-time operating system design and implementation, industrial communication systems and protocols, and formal methods for vulnerability and dependability analysis of distributed systems.

Stefano Scanzio was born in Italy in 1979. He received the Laurea and Ph.D. degrees in computer science from the Politecnico di Torino, Torino, Italy, in 2004 and 2008, respectively.

From 2004 to 2009, he was with the Department of Computer Engineering, Politecnico di Torino, where he was engaged in research on speech recognition and, in particular, he has been active in classification methods and algorithms. Since 2009, he worked with the National Research Council of Italy (CNR) and, currently, he is with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), Torino. He coauthored a number of technical papers in the area of computer science. His current research interests include communication protocols, industrial communication systems, real-time networks, and real-time operating systems.

Adriano Valenzano (SM'09) is Director of Research with the National Research Council of Italy (CNR). He is currently with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), Torino, Italy, where he is responsible for research concerning distributed computer systems, local area networks, and communication protocols. Since 1983, he has been involved in many national and international research projects and has led a number of research teams in the information and communication technology areas. He has coauthored about 200 refereed journal and conference papers in the area of computer engineering.

Dr. Valenzano was awarded as the coauthor of the Best Papers presented at the Fifth and Eigth IEEE Workshops on Factory Communication Systems (WFCS 2004 and WFCS 2010). He has served as a technical referee for several international journals and conferences, also taking part in the program committees of international events of primary importance. He served as a General Co-Chairman of the Sixth IEEE International Workshop on Factory Communication Systems (WFCS 2006), and took part in the Steering Committee of IEEE WFCS 2008 and WFCS 2010 and in the Advisory Committee of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) Series since 2008. Since 2007, he has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He is also Vice-President of the Piedmont Chapter of the Italian National Association for Automation (ANIPLA).

Claudio Zunino received the Degree in computer engineering and the Ph.D. degree in software engineering from the Politecnico di Torino, Torino, Italy, in 2000 and 2005, respectively.

Since 2006, he has been a Researcher with the National Research Council of Italy (CNR). He is currently with the Institute of Electronics and Computer and Telecommunications Engineering (IEIIT). He has authored and coauthored several papers in international journals and conferences in the area of wireless communication, Industrial Ethernet protocols, computer graphics, parallel and distributed computing, and scientific visualization. He serves as reviewer for a number of international conferences and journals.