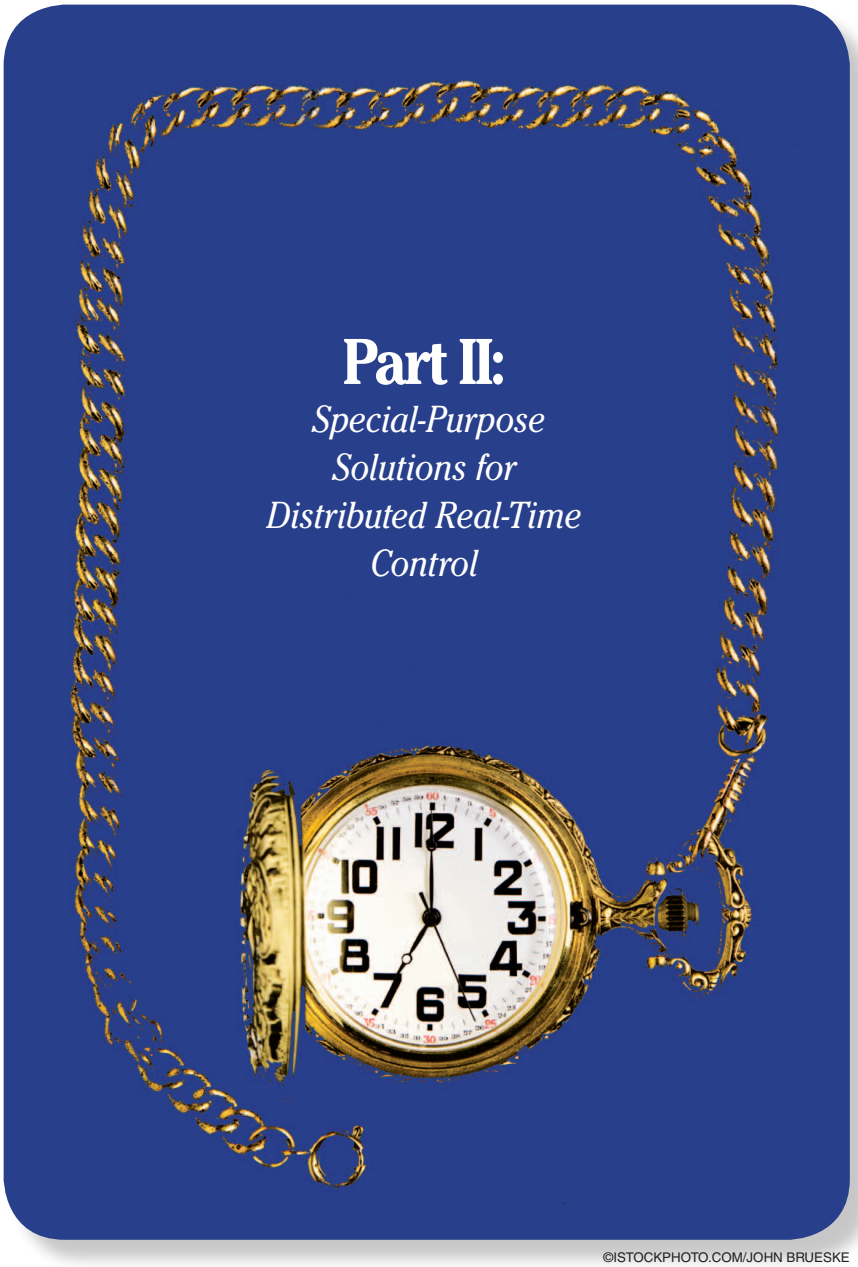
Synchronize Your Watches

GIANLUCA CENA,
IVAN CIBRARIO BERTOLOTTI,
STEFANO SCANZIO,
ADRIANO VALENZANO,
and CLAUDIO ZUNINO



©ISTOCKPHOTO.COM/JOHN BRUESKE

The ability to carry out coordered activities in distributed applications is currently considered a basic requirement in most industrial scenarios as well as in many areas with demanding real-time constraints, such as modern electric power systems, the automotive/avionic domains, and some types of networked embedded control systems. For this reason, the majority of the networks that were conceived recently for use in these environments provide some means to deal with distributed clock (DC) synchronization.

The capability to synchronize devices and applications accurately was already available in some fieldbuses. However, these kinds of networks typically have simpler mechanisms, wherein broadcast messages are used to trigger simultaneous actions on several nodes directly. This is the case of, e.g., the PROFIBUS [1] Sync and Freeze Modes or the Sync Object in CANopen, an application protocol based on controller area network (CAN) [2]. Such an approach is not as powerful as clock synchronization, since communication delays and jitters affect system accuracy directly. Moreover, the loss of a synchronization message usually leads to incorrect system behavior, since the related actions could not be triggered on devices.

The network time protocol (NTP) [3] and precision time protocol (PTP) [4] are two popular general-purpose clock synchronization protocols that are currently exploited in many

Digital Object Identifier 10.1109/MIE.2013.2248431
Date of publication: 17 June 2013

industrial systems. In particular, NTP is mostly used at the higher levels of the automation pyramid [5] (company), whereas PTP suits the needs of the lower levels (field). Applications belonging to the intermediate levels (cell) can employ either one, depending on the required accuracy. As highlighted in [6], NTP and PTP do not rely on specialized communication technologies. For this reason, they are able to provide high flexibility and enable interoperability between different platforms in heterogeneous systems.

Since timing constraints are the most important aspect at the field level, high-performance solutions are usually needed. When time-critical applications (e.g., motion control) are taken into account, a typical requirement is to keep synchronization errors strictly below 1 μs. In PTP—as in the majority of high-accuracy synchronization protocols—these errors mainly depend on the timestamping method. To obtain submicrosecond accuracy, timestamps must be acquired in hardware (h/w). State-of-the-art h/w-based timestamping approaches allow subnanosecond accuracy to be reached [7]. Better results can only be achieved by using modified physical layers, for instance, synchronous Ethernet (SyncE) [8], [9].

In addition, the approaches adopted in safety-critical networked control systems must also be very dependable. For example, PTP is typically used to synchronize intelligent electronic devices (IEDs) in today's electrical substations based on IEC 61850 [10], [11]. This is considered a critical task because both submicrosecond accuracy and high dependability are required (e.g., synchrophasors are used to sample electrical signals on a smart grid). Automotive x-by-wire applications [12]–[14] are another example of safety-critical networked control systems. In this case, precise synchronization enables error detection in the time domain—in addition to the mechanisms operating in the value domain, like the cyclic redundancy check—and ensures fail-silent behavior through bus guardians.

Several real-time Ethernet (RTE) solutions based on switched network topologies rely on PTP (or some modified version of it) to support clock synchronization. Very important examples of such networks are EtherNet/Internet protocol (IP) and, in particular, its common industrial protocol (CIP) [15], in which the CIP Sync object is defined to model the first version of PTP (i.e., IEEE 1588-2002), PROFINET IO [16], which is based on a modified version of IEEE 1588-2008 known as precision transparent clock protocol (PTCP), and the next version of Ethernet POWERLINK (EPL) [17], which will rely on IEEE 1588 [6]. A brief description of CIP Sync and PTCP is provided in [6]. Generally speaking, their behavior and performance are basically the same as those of PTP.

Other RTE solutions, particularly those adopting specific communication h/w, make use of ad hoc protocols instead to improve synchronization accuracy by typically taking advantage of some peculiar feature of the underlying network. Popular examples are Ethernet for control automation technology (EtherCAT) [18] and the latest version of the serial real-time communication system (SERCOS III) [19], which rely on a ring network topology, and EPL [17], which exploits the fact that the network is connected through hubs and repeaters.

The aforementioned approaches are appropriate for industrial systems because they rely on a centralized approach in which a single node (master) maintains the reference time and distributes it to all the other nodes in the network (slaves). They enable the creation of clear timing hierarchies in the network and, possibly, the synchronization with an external time source. Although absolute time sources—e.g., coordinated universal time (UTC)—are becoming a fundamental requirement in many application fields (e.g., energy distribution), there are several contexts that do not require synchronization to an absolute time. For example, industrial automation systems are typically closed environments. It is worth pointing out that energy distribution is also increasingly relevant also in field-level automation. Therefore, the need for an absolute time reference will likely arise in the near future in many industrial plants.

A distinct kind of application exists that only requires syntonization. In this case, only frequency is compensated, so that the clocks tick at the same speed, but their phases are not aligned. Conversely, synchronization protocols conceived for automotive/avionics networks, such as FlexRay (FR) [20] and time-triggered Ethernet (TTEthernet) [21], usually derive the reference time by means of a distributed approach. In this way, fault tolerance is improved noticeably with respect to master–slave solutions. The main drawback is that nodes only share a common relative view of time.

Mechanisms in networked control systems improve their robustness against communication jitters and reduce reliance on strict synchronization [22]–[24]. Such approaches can be profitably adopted to improve the quality of control over networks characterized by high variations in transmission times (IP channels, CAN buses, etc.). However, they can hardly be considered a replacement for synchronization protocols when time-critical applications are taken into account (motion control, x-by-wire, phasor measurement, etc.).

Three popular special-purpose synchronization protocols for wired networks are the DC mechanism of EtherCAT, the synchronization technique of EPL, and FR. The first two protocols can be considered to represent a wider class of centralized solutions for industrial environments, conceived to run on top of RTE. In particular, DC is a sophisticated mechanism that can measure and compensate propagation delays autonomously, whereas EPL is much simpler but requires some more effort in the setup phase. It should be noted that other solutions, such as the one adopted in SERCOS III, lie between DC and EPL. Like DC, propagation delays are evaluated at run time in SERCOS III,

although the synchronization scheme resembles EPL. In fact, accurate isochronous time marks are provided to devices through specific messages—i.e., master synchronization telegrams (MSTs)—that enable coordered actions but do not implement (in the initial version, at least) clock synchronization. FR is instead a meaningful example of fault-tolerant distributed synchronization protocol.

A comparison between these protocols and general-purpose solutions (NTP, PTP, and PTP-based approaches) is carried out at the end of the article, taking into account the main aspects that are relevant to control systems.

EtherCAT DC

EtherCAT [18] is a high-performance Ethernet-based industrial network. Its main goal is to enable the adoption of Ethernet communications in automation applications that require short cycle times and low communication jitters. This protocol is an open standard based on master–slave architecture. An important feature of EtherCAT is the DC synchronization mechanism, which enables all devices to share the same system time. In this way, typical control operations, such as the generation of coordered output signals or the precise timestamping of input events, can be synchronized for all devices in the network.

Communication Infrastructure

EtherCAT networks are based on an open physical ring topology connecting all the nodes, as shown in Figure 1. Since standard Ethernet frames are used, a conventional network controller can be employed for the master station, and no special h/w is required. A PC provided with a full-duplex Ethernet board is frequently used for the purpose.

At the bottom of the protocol stack, EtherCAT supports two different kinds of physical layers, namely Ethernet and E-BUS. The former is used to set up a network according to the traditional Ethernet specifications. A typical use is the connection of the master to the network segment(s) including the slaves. Conversely, E-BUS can be used only as a backplane bus. In particular, its physical layer is designed to reduce pass-through delays inside the nodes. From a global point of view, an EtherCAT segment can be thought of as a single, distributed device that is connected to the controller and is able to receive and send Ethernet frames. This device, however, includes a (possibly large) number of EtherCAT slaves.

An important feature of EtherCAT is the DC synchronization mechanism, which enables all devices to share the same system time.

The master interacts with the slaves by using EtherCAT commands. Such commands are encoded through telegrams contained in conventional Ethernet frames. A number of telegrams can be encapsulated in the same frame to reduce the overheads when small-sized process data have to be exchanged. In this way, the protocol can achieve a very high throughput. Telegrams are processed on the fly by the slave nodes; this is another reason for the very high performance of this protocol. Since on-the-fly processing capabilities require purposely designed EtherCAT slave controllers (ESCs) [25], standard Ethernet equipment cannot be used as slaves.

As mentioned previously, the DC mechanism enables all the devices in the network to share the same system time. In particular, all slaves are synchronized to the same reference clock. DC is placed above the EtherCAT access protocol, and its implementation is not mandatory. For this reason, both DC-enabled and non-DC-enabled devices can quietly coexist in the same network. It is worth noting that DC is not a general-purpose synchronization protocol, since it relies on specific features of EtherCAT, such as its ring topology and the on-the-fly processing capability for telegrams.

To better understand the DC mechanism, let us introduce the following definitions.

- The system time is a 64-b value representing the elapsed time in nanoseconds since 0:00 h of 1 January 2000.
- The reference clock is an EtherCAT device that is responsible for providing the time reference. Usually, the slave with DC capability closest to the master along the ring is selected for this purpose.
- Each DC device has a local clock that, if not suitably controlled, runs independently of the reference clock.

Synchronization Mechanism

The clock synchronization process (CSP) consists of the following three main actions:

- *Propagation delay measurement:* After collecting specific timestamps from all the slaves, the master, which is aware of the network topology, computes the propagation delay for each segment.
- *Offset compensation:* The master computes the offset between the

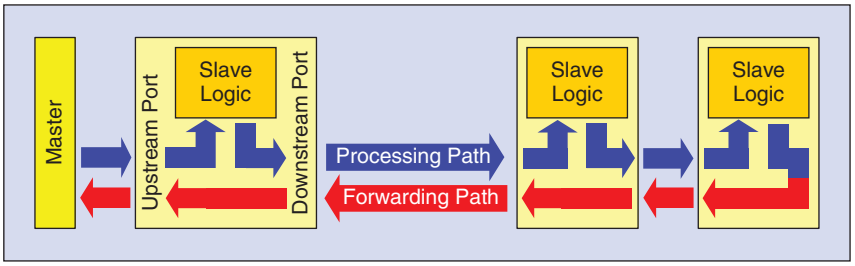


FIGURE 1 – The EtherCAT network architecture.

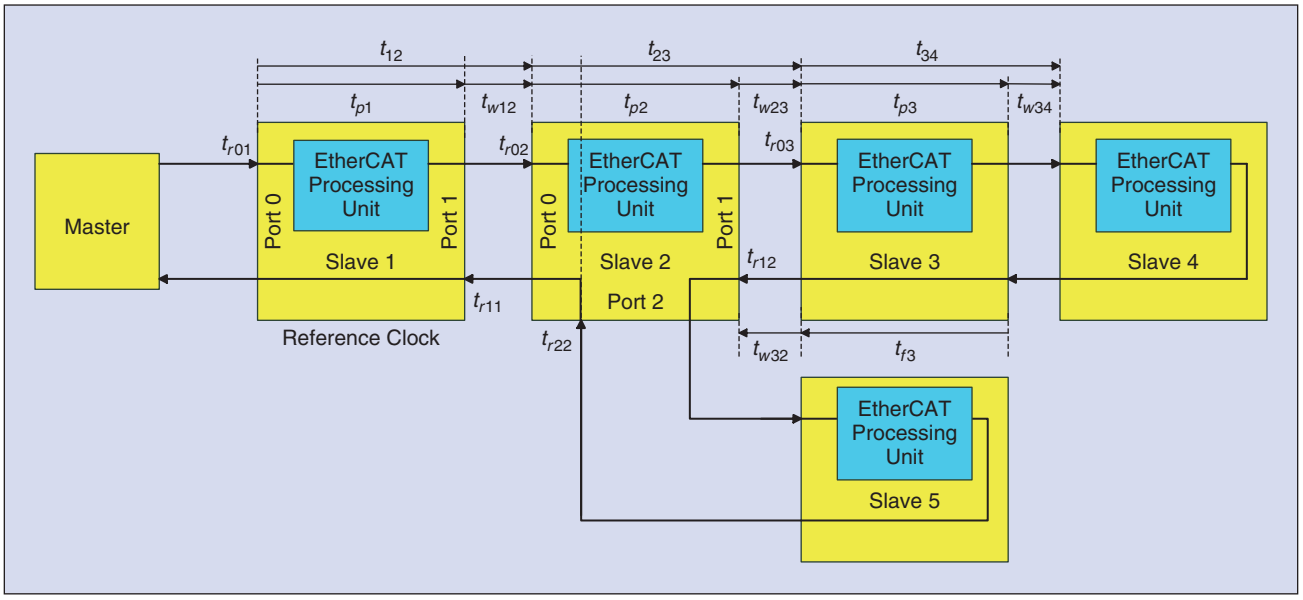


FIGURE 2 – Propagation delay computation.

reference and local clocks separately for each DC-enabled slave. The difference is then compensated individually by writing the suitable correction value into a specific register of each slave. When this step is completed, all devices share the same absolute system time.

- *Drift compensation:* Drifts are compensated by regular measurements of differences and readjustments of the local clocks.

The presence of a slave node causes small processing/forwarding delays, both inside the device and over the communication medium. For this reason, the propagation delay between the reference node and the slave has to be evaluated carefully. This procedure consists in turn of three main steps. First, the master sends a broadcast message (i.e., a telegram executed by every slave). Second, each slave stores the value of its local clock when the first bit of that message is received. This action is performed for each port of the device (i.e., on both the processing and forwarding paths in Figure 1). Finally, the master collects the timestamps and computes the path delays, taking into account the topology of the network. During this procedure, the slave devices do not need to be synchronized because only local clock values are used.

The computation of each propagation delay ($t_{\text{PropagationDelay}}$) is based on the differences between the receiving times at the device ports. Moreover, delays introduced by nodes without DC capabilities are treated as additional wire delays. Note, however, that devices equipped with more than two (bidirectional) ports are compelled to support the measurement of propagation delays. Figure 2 shows a sample EtherCAT network consisting of four slaves with two ports and one device with three ports. The node closer to the master (slave 1) is selected as the reference clock. Computation of propagation delay uses the main elements summarized in Table 1. The algorithm starts when the master sends a set of telegrams to reset the slaves' DC registers. After this initialization phase, the master transmits a pair of telegrams to each slave: the first message forces the device to record timestamps related to the frame, while the second is used to read the timestamps back. As mentioned previously, timestamps are taken in both directions, i.e., a first timestamp is acquired when the message is received on the processing path, while a second timestamp is recorded on the reception of the same frame on the forwarding path.

In Figure 2, for instance, slave 2 records the timestamp t_{r02} when the frame reaches its port 0, t_{r12} when the frame is received back from slave 3 and, finally, t_{r22} when the message is returned by slave 5. Since timing information is referred to the same internal clock, the difference between each pair of timestamps can be evaluated properly. Differences are then used to evaluate all delays (processing, forwarding, and wire contributions) and compute the resulting propagation delay for the slave. The forwarding delay for the first slave (slave 1 in Figure 2) cannot be computed, since timestamps are recorded only for the incoming messages. A possible solution is to assume that it is the same as the other t_{fx}. Alternatively, it can be measured once and for all, at least in theory, when the device is designed and manufactured. The compensation is then performed directly by the

TABLE 1—PROPAGATION DELAY PARAMETERS.

t_{px}	Processing delay of slave x
t_{fx}	Forwarding delay of slave x
t_{xy}	Propagation delay from slave x to slave y
t_{wxy}	Wire propagation delay between slaves x and y (perfect symmetry assumption)
t_{rky}	Receive time on port k of slave y, recorded with a write access to the DC receive time register

master, likely by using a predefined fixed value.

The second DC action is the offset compensation. Local clocks ($t_{\text{LocalTime}}$), when not synchronized, resemble free-running counters. At system start-up, the clocks do not use the same value as the reference clock, so an offset compensation is needed. The technique adopted for this purpose in EtherCAT is based on the same timestamps introduced in the previous step. In fact, after timestamps have been collected, the master is also able to evaluate the offset of each local clock with respect to the reference time (t_{Offset}). This value is then written into a suitable register (system time offset) of the slave device and then used to adjust the local time. As a consequence, after the initialization phase, each DC slave can determine its own copy of the system time autonomously by using its local time and offset values.

The last action in DC is the drift compensation. Natural drifts of local clocks (due, for instance, to oscillator tolerances) are corrected by means of a time control loop (TCL) algorithm implemented right into each EtherCAT slave controller. The master periodically distributes the system time ($t_{\text{ReceivedSystemTime}}$ as read from the reference clock) to all the slaves in the network. The TCL algorithm in each slave compares the received system time to its local copy.

In evaluating the difference Δt, propagation delays must also be taken into account:

$$\Delta t = (t_{\text{LocalTime}} + t_{\text{Offset}} - t_{\text{PropagationDelay}}) - t_{\text{ReceivedSystemTime}}. \quad (1)$$

If Δt is positive, the local time is running faster than the system time and needs to be slowed down. On the contrary, when Δt is negative, the local time is too slow and needs to be sped up.

TCL is responsible for adjusting the local clock speed. Slave devices should nominally increment the local time by ten units every 10 ns. As a result of the comparison mentioned previously, however, the real increment may be equal to either 11, ten, or nine units, depending on the value of Δt. In this way, the monotonic property of time is assured for each DC-enabled slave.

To smoothen abrupt changes and prevent oscillations of the Δt value, the mechanism is further controlled through a filtering procedure. In particular, when the local clock is too fast, a typical series of values used to compensate Δt might be (10, 10, 10, 10, and 9) ns, whereas, when the local clock is too slow, the sequence would likely be (10, 10, 10, 10, and 11) ns. After the initialization of delays and offsets, the master tries to compensate the static clock deviations quickly, i.e., by sending a high number of commands (e.g., 15,000) in separate frames. In other words, first the control mechanism takes care of the static deviations to synchronize local clocks, then compensation frames are sent cyclically to correct dynamic clock drifts.

Recent Advances

Synchronization to an external time source has been recently introduced in EtherCAT. In this way, the time alignment of separate EtherCAT segments is now possible. The basic idea is to control the DC time using an external device. A possible solution relies on the availability of a specific device (e.g., Beckhoff EL6692 [26]), which provides a common DC reference time to different network segments. In practice, the E-BUS is cross-connected to an Ethernet segment, and the user has to decide which side (either E-BUS or Ethernet) hosts the reference clock with the higher priority. The connecting device will then be responsible for sending a reference clock correction value to the EtherCAT master with the lower priority. In addition, when the reference time in an EtherCAT system has to be adjusted to a higher-level clock, an external synchronization device can be used that supports a synchronization protocol different from DC. Terminals exist (e.g., Beckhoff EL6688 [27]) that enable the adoption of a PTP external clock source. In principle, several other popular synchronization protocols and clock sources can also be used, e.g., global positioning systems, radio clocks, NTP, and SNTP.

To cope with systems that demand increased dependability, an optional redundancy mechanism has been defined as well, which both enables devices to be replaced without having to shut down the network and makes the system resilient to slave failures. In this case, a specific slave device (e.g., Beckhoff CU2508 Ethernet-Port-Multiplier) has to be used, which behaves as the reference clock for network segments.

Performance

Several articles in the literature have dealt with EtherCAT and its performance, but only a few of them have focused on DCs. In general, DC achieves good precision (several tens of nanoseconds), as confirmed by the experiments presented in [28] for real-world devices. In that study, the accuracy of the DC mechanism was evaluated for different network configurations. In the considered situations, the authors showed that accuracy is not significantly affected by either the network size or the number of devices. In [29], the real network traffic was also analyzed by logging all the messages exchanged in the different phases of the DC initialization. Results show that, in real systems, some additional tricks are implemented to enhance the performance.

Some articles have also addressed the problem of the synchronization (or lack thereof) between master and slave devices. For instance, [30] proposes a master–slave synchronization technique for a force measurement system, while [31] describes a robust solution, based on EtherCAT, to offer high-redundancy capabilities in combination with accurate time synchronization.

Ethernet POWERLINK

EPL is a hard real-time industrial network that, unlike EtherCAT, was conceived to use unmodified Ethernet equipment. In its current version [17], [32], EPL does not rely on IEEE 1588 for synchronization. On the

Isochronous transmission of frames is supported by the POWERLINK protected mode cycle structure.

contrary, it defines its own mechanism to achieve the best accuracy for the specific network configuration on which it is based. EPL belongs conceptually to the same class of synchronization protocols as DC (centralized master–slave, conceived to run over RTE, and not based on PTP) and shares many similarities with it, including high accuracy. Nevertheless, it is noticeably simpler than DC, mainly because it does not provide any mechanism to evaluate propagation delays dynamically, and not even true clock synchronization (it just supports isochronous operations).

Communication Infrastructure

EPL relies on plain Ethernet. To ensure accurate timings, low-jitter repeaters and hubs should be used instead of switches, at least in the current version. Star, line, and tree topologies are allowed. Collisions, typical of half-duplex Ethernet, are completely prevented in EPL protected segments thanks to a specific mechanism, called slot communication network management (SCNM), which lies just above the legacy Ethernet and resembles the well-known producer/consumer/arbiter approach.

Each EPL segment has exactly one managing node (MN), which is responsible for coordinating network access of all the other nodes—also known as controlled nodes (CNs). Cyclically (and with a very low jitter, well below 1 μs), the MN executes a POWERLINK cycle. Each cycle begins with the broadcast transmission of a start-of-cycle (SoC) frame, which also denotes the beginning of the isochronous phase. Then, every CN is queried separately by means of a PollRequest frame (PReq). The CN has to reply with a PollResponse (PRes) sent in multicast to enable the producer/consumer paradigm. The cycle ends with an asynchronous phase to support sporadic transmissions too.

Needless to say, all nodes must obey the SCNM rules so as to ensure deterministic communication and maintain hard real-time behavior. For this reason, suitable gateways are requested to enable IP communication between nodes on a protected EPL segment and other IP-based networks (e.g., the Ethernet backbone), which are known as type 1 EPL routers. The aim of these routers is to prevent non-EPL messages from affecting the EPL cycle by slotting TCP traffic properly.

Synchronization Mechanism

Isochronous transmission of frames is supported by the POWERLINK protected mode cycle structure. Since each EPL protected segment is basically a broadcast domain on which collisions cannot occur, synchronization in EPL is accomplished in a straightforward way. The SoC frame, which is broadcast by the MN at the beginning of every POWERLINK cycle, is taken as the basis for the common timing of all CNs. Because of the medium access rules, this frame cannot be delayed by the other nodes. When receiving the frame, every CN is enabled to carry out highly accurate network-wide coordinated data acquisition and actuation.

In addition, the SoC frame may optionally contain, encoded in the NetTime field, the network time as seen by the MN. It is expressed as an absolute time and includes the number of seconds and nanoseconds elapsed from 1 January 1970 at 00:00 h. Although there is no explicit mention of this in the EPL specifications, when receiving the SoC frame, each CN could, in theory, adjust its local clock consequently.

Recent Advances and Performance

As discussed previously, plans exist to replace hubs by switches in the next version of EPL and to use PTP for clock synchronization. Synchronization accuracy in the existing version of the protocol basically depends on the jitter with which SoC frames are sent by the MN, which can be improved noticeably by using customized implementations (e.g., based on field-programmable gate arrays). Moreover, EPL does not compensate propagation delays automatically. On the contrary, every CN is enabled to do so by using a static approach by configuring a specific parameter ($t_{\text{propag\_CNn}}$) during the network configuration phase. Overall, accuracies below 100 ns are attainable, according to the Ethernet POWERLINK Standardization Group (EPSG) [33].

FlexRay

FR is a communication protocol developed for automotive systems to enable interconnection of electronic control units in vehicles. It was explicitly conceived to overcome a number of drawbacks of existing solutions, in general, and of the CAN protocol, in particular. Thanks to several factors, such as the time-triggered medium access technique, the native availability of a redundant (double) physical channel and bit rates as high as 10 Mb/s, FR achieves noticeably higher degrees of determinism, fault-tolerance, and performance than CAN.

From a conceptual point of view, FR builds on the same mechanisms developed for Byteflight [34] and class C time triggered protocol (TTP/C) [35] (it likely adopts the best features from both these protocols). With respect to pure time-triggered protocols, such as TTP/C, FR offers much higher flexibility, mostly because of its ability to manage dynamically asynchronous data exchanges through a distributed prioritized medium access mechanism.

Until 2009, specifications were managed by the FR Consortium, which included most of the leading car manufacturers from all over the world. The consortium was closed after finalizing version 3 of the specifications. The following description is based on [20]. This document is basically an extension of version 2.1, which has been adopted for the design of several FR controllers.

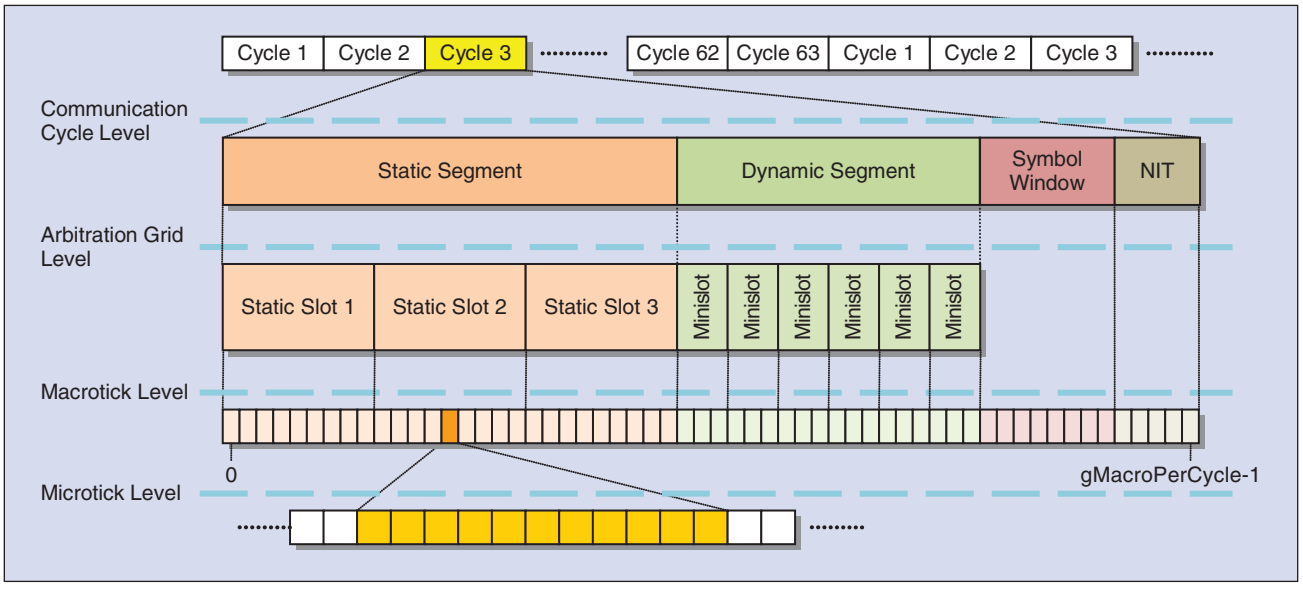


FIGURE 3 – The FR protocol timing hierarchy.

Since FR can be seen as the successor of CAN for in-car applications, nothing prevents—from a technical point of view—its adoption for industrial applications as well [36]. However, its use outside the automotive scenario was not envisaged in any way by the FR Consortium. Consequently, its expected penetration level in factory automation environments is currently quite low.

Communication Infrastructure

The communication subsystem of FR is based on a peculiar medium access control (MAC) technique, which affects most of its properties directly. Access to the shared transmission support is based on a communication cycle that is repeated indefinitely, as shown in the upper part of Figure 3. The duration of the cycle is fixed, though configurable.

Each communication cycle is divided into segments (up to four), namely the static and dynamic segments, the symbol window, and the network idle time (NIT). In the (mandatory) static segment, a conventional time division multiple access (TDMA) mechanism is adopted. As each transmitting node is assigned its own time slot, collision-free communication is enabled, which takes place with high determinism. In the (optional) dynamic segment, a special technique known as a flexible time division multiple access (FTDMA) is used. FTDMA relies on a minislotting approach in which messages are characterized by consecutive identifiers. Each node that does not have a message to transmit generates a period of inactivity on the network (minislot). The duration of a minislot is much shorter than the static slot length, enabling flexible data exchanges, whose schedule is decided at the run time. The (optional) symbol window can be used to exchange a single symbol and its arbitration is not managed directly by FR: it is mainly used in the startup phase. Finally, NIT is a period at the end of each cycle when the network is kept idle. NIT is used to perform clock corrections.

Precise synchronization of all nodes in the network is a prerequisite for the proper operation of the TDMA and FTDMA mechanisms. For this reason, a timing hierarchy is defined that consists of four levels, as depicted in Figure 3. The arbitration grid level is placed just below the topmost communication cycle level. This is where the static segment is split into a fixed (configurable) number of (static) slots of the same duration. Each slot is assigned to a specific node (though the same node can own more than one slot) and can be used to send exactly one frame. Such an assignment is carried out in the configuration phase, before the system is started, and cannot be changed during the normal network operation. The dynamic segment, in turn, consists of several identical minislots. Each frame takes an integral number of minislots for its transmission, so messages of different sizes can be accommodated easily. This technique permits the spontaneous transmission of sporadic frames when needed by a device. The medium access technique in this segment is based on message priorities and the resulting behavior someway resembles CAN (but the actual mechanisms of the two protocols are very different).

The macrotick level is found beneath the arbitration grid. At this level, every node sees the time as a sequence of macroticks with the same duration. Precise alignment of macroticks is maintained by means of a suitable synchronization mechanism. Action points are also defined here: they are global time instants when significant events must/can occur. The concept of global time in FR is quite different from other popular solutions. In this case, in fact, no reference node is present and a true, absolute time is not defined. Nevertheless, at any instant, all nodes share a common view of both the current cycle (vCycleCounter parameter) and macrotick (vMacrotick parameter)

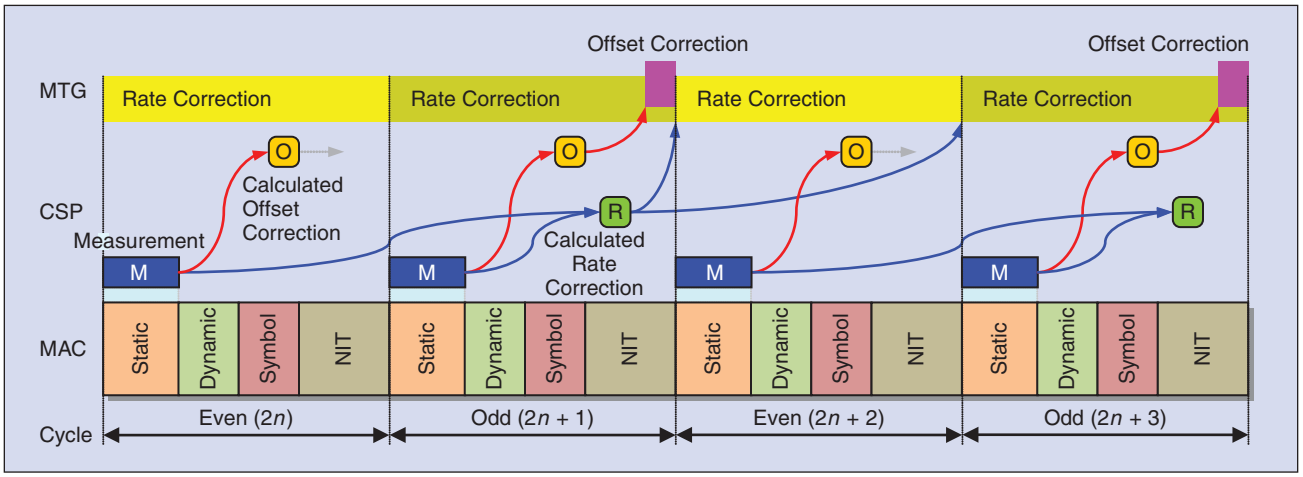


FIGURE 4 – Clock synchronization in FR.

in the cycle. This can easily be exploited by the upper levels to perform coordinate actions, thus providing the same kind of behavior enabled by the global time.

Microtick is the lowest level in the hierarchy. Every node generates microticks from the local oscillator, usually through a prescaler. Microticks in the different nodes neither have the same duration nor are synchronized in any way. This level is mainly related to practical controller implementations.

Synchronization Mechanism

Unlike most synchronization mechanisms, such as DC, NTP, and PTP, which are placed on top of the communication layer, communication and synchronization in FR are tightly interleaved. In fact, during the normal operation (i.e., after the startup phase), correct FR transmissions require proper synchronization of all the nodes in the network. At the same time, the synchronization mechanism is based on message exchanges between the nodes. It is worth noting that such an approach was already adopted successfully in other hard real-time communication solutions such as TTP/C. In particular, a lot of work was done on TTP/C to prove its correct behavior and temporal and dependability properties.

Clock synchronization in FR basically relies on two processes that operate concurrently, namely the macrotick generation (MTG) and the CSP (see Figure 4). MTG grants the alignment of macroticks (and, consequently, of the arbitration grid) across the whole network by applying suitable rate and offset corrections. In contrast, CSP is responsible for both measuring deviations of the local clocks and computing values to correct their rate and offset.

Concerning the CSP, FR nodes measure the difference between the expected and the actual arrival times for every sync frame exchanged in the static segment. The expected arrival time is a static slot action point, whereas the actual time is the instant when the frame is received. Timestamps are obtained through the local oscillator and are expressed in microticks.

Sync frames are a subset of all static frames. In fact, only packets sent by nodes equipped with high-quality oscillators are marked sync. In version 2.1 of the protocol, every node was allowed to send at most one sync frame per cycle, but this restriction has been relaxed in version 3. A minimum of three sync frames per cycle are needed when a fault-tolerant behavior has to be ensured, otherwise two sync frames are enough to let the mechanism work properly.

Nodes store deviation values (for both channels and separately for even- and odd-numbered cycles) in local tables and then execute a fault-tolerant midpoint (FTM) algorithm.

1) A suitable value k is selected based on the number of rows in the table (e.g., $k = 0$ for one or two rows, $k = 1$ for three to seven rows, and $k = 2$ otherwise).
2) The list of measured values is sorted, and the k smallest and largest values are discarded.
3) The smallest and largest values in the remaining set are selected, and their average computed.

The value obtained from FTM is assumed as the deviation of the local clock from the global clock. For instance, if the ordered list of deviation values is $(-10,-7,-3,+1,+4,+5,+12,+16)$, which implies $k = 2$, the values -10, -7, $+12$, and $+16$ are first discarded, and then the average between -3 and $+5$ is computed to obtain a deviation equal to $+1$. The estimation of the offset deviation occurs on every cycle, whereas the rate deviation is evaluated once for every pair of consecutive cycles. Indeed, a pair consists of an even- and an odd-numbered cycle, in that order. The rate deviation determined in any pair is used to correct the oscillator rate in the following cycle pair. Furthermore, computed values are checked against suitable limits before applying corrections. If the values exceed the predefined limits, suitable recovery procedures are undertaken.

The MTG process adjusts the local MTG rate by tuning a parameter that specifies the number of microticks per cycle. This mechanism has been

conceived so that corrections can be applied smoothly and abrupt changes in the local view of time avoided. It is worth noting that rate correction is carried out in every cycle, whereas offset are corrected by enlarging (or shortening) the NIT segment in odd-numbered cycles only. In particular, rate-correction parameters are varied when entering the offset-correction phase. Figure 4 shows the relationship (and order) between the measurement of deviations and the points in time when correction values are calculated, together with the way they are used for aligning clocks.

Network start-up deserves some more explanation. In FR, no synchronization is possible without communication and vice versa. This apparent deadlock is solved by means of a suitable startup procedure. Some selected sync frames, in fact, are responsible to act as startup frames. Startup frame senders are known as coldstart nodes. When the network is booted, coldstart nodes detect that no valid transmission is taking place and begin the coldstart procedure. In this phase, possible collisions among coldstart nodes are solved. The mechanism also assures that, after a certain time has elapsed, one leading coldstart node initiates the communication cycle successfully, whereas all the other nodes simply join the cycle. The FR specifications also include a second class of devices, which are not able to behave as coldstart nodes. This means they have to wait until someone else completes the startup sequence, before being able to join the steady-state communication cycle.

Recent Advances

In the above conditions, direct master–slave synchronization is not possible. In the same way, the protocol does not provide any mechanism to allow the synchronization of subordinate clusters (either buses or rings) to a main cluster. Both these characteristics, indeed, reduce the degree of reliability to some extent; however, they can be exploited to decrease costs and enhance composability. FR version 3 offers some more support in this direction. In particular, two additional synchronization modes (TT-L and TT-E) have been added to the basic mechanism (TT-D).

The local sync mode (TT-L) is used in very simple networks. In this case, a single node (TT-L coldstart node) is enabled to behave as a startup and synchronization device by sending two sync frames per cycle. The external sync mode (TT-E), instead, is used in multicluster networks. Subordinate clusters are synchronized to the primary TT-D group by means of a gateway (TT-E coldstart node). If the TT-D cluster is not active, the system reverts to the local sync mode. In both cases, synchronization is transparent to nonsync nodes, which means that backward compatibility is preserved.

Performance

The synchronization mechanism of FR features both high accuracy and a very good degree of fault tolerance. The latter characteristic is mandatory to satisfy the tight safety constraints imposed by control applications in the automotive domain such as in x-by-wire systems. As long as most nodes are working correctly, FR is able to ensure their accurate synchronization. This means that no single point of failure can affect synchronization (and, consequently, communication). The most important performance index for the protocol is then its ability to preserve the correct behavior also in case of node failures.

FR is a complex protocol that makes use of many operating parameters, which, in turn, directly affect its performance [37]. Generally speaking, precisions in the order of several tens of nanoseconds can be obtained. Studies presented in [38], [39] show that, under realistic conditions, the system precision stays in the order of a few hundreds of nanoseconds, even when the frequencies of some local oscillators deviate from their nominal values in a nonnegligible way.

Comparison

Basically, all clock synchronization protocols rely on similar techniques (time measurement, propagation delay evaluation, offset, and rate compensation), differing mainly in the way the mechanisms are implemented. The following comparison considers several protocols: besides the special-purpose mechanisms described here (DC, EPL, SERCOS III, and FR), the NTP and PTP general-purpose solutions have been taken into account as well (see [6] for more details). RTE solutions based on PTP have not been dealt with explicitly because they resemble PTP closely. The comparison is carried out from several points of view but focuses, in particular, on their use in industrial and embedded control systems. The main results are summarized in Table 2.

Underlying Network

The first, main difference concerns the type (and size) of underlying network. NTP was primarily conceived for geographic heterogeneous networks, so it relies directly on the user datagram protocol (UDP) and IP. For this reason, it is also the most frequently adopted solution for synchronizing clocks of computer systems interconnected through the Internet. Currently, many popular operating systems (e.g., Linux, Windows, and Mac OS X) provide native (and free-of-charge) NTP support.

PTP, on the contrary, was mainly designed with local networks in mind. Here, the term local does not necessarily mean small, as they may easily stretch over several kilometers—e.g., they are able to cover a whole industrial plant. Although PTP does not mandate any specific kind of network, the master is required to efficiently distribute the synchronization messages to all the related slaves. As a matter of fact, this does not make the protocol very suitable for geographic networks. PTP is likely the most popular solution for enabling accurate synchronization in Ethernet networks, and many inexpensive off-the-shelf devices (i.e., switches and network interface cards) are currently available that support this protocol as well as some open-source implementations [40].

By targeting smaller and more homogeneous networks, a noticeably

TABLE 2—COMPARISON OF SYNCHRONIZATION PROTOCOLS.

PROTOCOL	NTP	PTP	DC	EPL	FR
Specification	IETF RFC 5905	IEEE 1588	IEC 61158 CPF 12	IEC 61158 CPF 13	FR communication system
Last version	2010 (Version 4)	2008	2010 (Ed. 2.0)	2010 (Ed. 2.0)	2010 (Version 3.0.1)
Synchronization	Clock synchronization	Accurate clock synchronization	Accurate clock synchronization	Accurate isochronous synchronization	Accurate synchronized time grid
Network	Any with UDP/IP support	Any with multicast	Only EtherCAT	Only EPL	Only FR
Extension	Geographic	Plant	Shop floor	Shop floor	Vehicle
Time reference	UTC	Centralized	Centralized	Centralized	Distributed
Technique	Client–(multi)server	Hierarchical master–slave	Ring-based master–slave	Broadcast-domain master–slave	Producer–consumer self-aligning arbitration grid
H/W support	Not foreseen	Optional	Mandatory	Optional	Mandatory
Quality	ms	(Sub-)μs	Sub-μs	Sub-μs	Sub-μs
Dependability	Several servers can be set up	BMC, alternate master, grandmaster clusters	Double ring and floating reference time	Redundant MN	Never-give-up strategy, FTM

better accuracy than is possible with NTP is usually obtained. For this reason, PTP is currently considered a flexible solution for high-precision synchronization needs and is often adopted, possibly in some modified versions, in many RTE networks EtherNet/IP, PROFINET IO, etc.). Moreover, it is seen more and more as the glue for achieving synchronized behavior across distinct systems—including those based on different communication technologies. For instance, nodes on a number of separate EtherCAT segments can be synchronized by using suitable devices known as IEEE 1588 external synchronization interfaces [27], which are interconnected through a PTP-enabled Ethernet infrastructure.

Special-purpose synchronization solutions are far less general than NTP and PTP. For example, DC can be used only on top of EtherCAT networks. Although EtherCAT systems may actually stretch over large areas, RTE networks are basically intended for use on the shop floor of industrial plants, and their size rarely exceeds a few hundred meters. Several other RTE networks exist that define their own synchronization mechanism to increase performance and reduce complexity, e.g., EPL and SERCOS III. The common aspect in all these cases is that synchronization relies on peculiar features of the underlying network. For example, DC and SERCOS III exploit the ring network topology to evaluate propagation delays, whereas EPL requires that the underlying network be a collision domain to broadcast timing marks.

Unlike the other solutions that are disjointed from the communication subsystem, the FR synchronization technique is strictly interleaved with the MAC mechanism. Indeed, one may not work in the absence of the other. FR is currently tailored to in-vehicle applications only, so the resulting systems are usually quite small in size (tens of meters).

Synchronization Technique

The protocols considered in this comparison adopt different approaches to achieve synchronization of local clocks. Typical NTP implementations are based on a client–(multi) server architecture, wherein clients send requests for the current time to one or more servers placed elsewhere. PTP adopts instead a centralized hierarchical master–slave approach, wherein the time is kept by the grandmaster clock. In every PTP subnetwork, a master is defined, which, in turn, takes care of providing the current time to all the subordinated nodes (from the point of view of synchronization, they behave as slaves).

DC, SERCOS III, and EPL are centralized master–slave solutions too, but the network may include only one segment. Subsegments can be possibly envisaged in some cases, which are connected through repeaters operating at the physical layer local area network (LAN) switching is not allowed for performance reasons). Unlike PTP-based solutions, slaves cannot take the initiative in the communication in these cases, since the underlying networks rely on the master–slave paradigm. This aspect affects noticeably the way the synchronization technique operates.

By measuring the propagation delays dynamically, very high accuracy can be achieved in DC with little effort in the setup phase. The same holds for SERCOS III. Synchronization in a single EPL-protected segment is based more or less on the same approach, but delay compensation has to be configured statically. In summary, the DC, SERCOS III, and EPL mechanisms actually work in a quite different way from PTP.

Finally, FR is a fully distributed solution, wherein a subset of the nodes generates a synchronized time grid for all devices in the network. The exchange of timing information, in this case, resembles the producer–consumer paradigm.

H/W Support

Accuracy also depends on the fact that the synchronization mechanism relies on specific h/w. In this respect, several cases can be distinguished,

depending on whether h/w support is not foreseen, optional, or mandatory. In cases where it is an option, different accuracies have to be expected for cases in which such support is actually available and those in which it is not.

Because of its intended application fields, NTP basically does not foresee any kind of h/w support. Although not strictly forbidden, specific NTP h/w is seldom used in real devices, if at all. Concerning PTP, though no specialized h/w is mandated, its use is practically unavoidable when dealing with applications with tight timing requirements (e.g., motion control). For this reason, almost all RTE solutions that rely on this protocol (CIP Sync) or its derivatives such as PTCP (PROFINET IO) for synchronization require some kind of h/w support. A big advantage of PTP is that the only operation that has to be carried out in h/w (or through h/w-assisted approaches) to achieve high accuracy is timestamping on frame transmission and reception. Currently, the number of Ethernet communication controllers and infrastructure components that provide IEEE 1588 support is increasing steadily.

In most special-purpose synchronization protocols, such as DC and FR, h/w support is mandatory. Some of these solutions, e.g., EPL V2, are in theory able to operate even in the absence of specialized h/w. However, doing so usually worsens the accuracy noticeably.

Timing Hierarchy

For a number of reasons, real-world control systems in which devices are interconnected through a communication network might demand accurate synchronization of several distinct subsystems to an external time source. This requirement mainly concerns power grids, and it is often found in large enterprise networks as well. However, the shop floor in industrial systems, as well as some networked embedded control systems, may exhibit similar needs to coordinate operations on distinct parts of the equipment.

On the contrary, DC, SERCOS III, EPL, and FR, which were purposely designed for control applications, can easily reach submicrosecond accuracy.

Of course, the implementation of the timing hierarchy can be significantly different for each protocol. The layered architecture of NTP, which relies on several logical levels (strata), deals with this aspect natively. The same is the case with PTP and its hierarchical physical network architecture, which is based on a master–slave approach. It is worth noting that the shop floor in industrial networks may have linear topologies that may include many cascaded devices. In these cases, solutions based on transparent clocks like PTCP are able to offer noticeably better accuracy by reducing the depth of the timing hierarchy and, consequently, synchronization errors.

In the EPL specification, a device known as the POWERLINK router is envisaged that can optionally behave as a boundary clock. As such, it can synchronize an EPL-protected segment to an external time source (e.g., PTP). Although the basic DC mechanism does not foresee such a feature explicitly, a group of EtherCAT master nodes are nevertheless allowed to synchronize using another protocol. In addition, slave devices of a special type have been introduced recently that are able to synchronize distinct EtherCAT segments. Similarly, a layered architecture was not envisaged in the original FR specifications, but version 3 overcomes this limitation by allowing the synchronization of different segments.

Quality of Synchronization

The network type and size, as well as the synchronization technique, affect the quality of synchronization. Accuracy and precision are frequently adopted to quantify this characteristic [41]: accuracy can be considered a measure of the average offset between local clocks and the reference time in steady-state conditions (residual offset), whereas precision indicates how much local clocks may deviate with respect to the reference time at any given point in time (jitters).

A detailed analysis falls outside the scope of this article. However, NTP appears unable to ensure the same synchronization quality as the other solutions. Clock deviations are typically in the order of milliseconds, though they can be noticeably shorter in intranets. This is not surprising, since NTP was developed to work in large heterogeneous networks. On the contrary, DC, SERCOS III, EPL, and FR, which were purposely designed for control applications, can easily reach submicrosecond accuracy.

PTP performance mainly depends on the actual network size, number of switches, h/w support, and propagation scheme (boundary versus transparent clock [4], [6]). Recent solutions that rely on transparent clocks (e.g., PTCP) allow synchronization performance in the order of tens of nanoseconds for networks based on star or tree topologies. This makes PTP-based solutions suitable for distributed control applications with demanding timing constraints and explains why PTP forms the basis for synchronization in many RTE networks.

Dependability and Flexibility

Dependability and flexibility are somehow related concepts, although they are clearly distinct. Generally speaking, all solutions include mechanisms to guarantee satisfactory levels for them. NTP clients can be configured to get the time from more than one server. In this way, synchronization is not disrupted in case the reference node becomes unavailable. Moreover, no explicit reconfiguration is needed to cope with this undesired event.

A best master clock (BMC) mechanism included in PTP permits the best time source in the system (grandmaster clock) to be selected at run time. Besides increasing flexibility, this approach also improves dependability because automatic replacement of the reference clock is possible in case of failures. Unfortunately, the BMC may take exceedingly long times (longer than acceptable for some kinds of application).

To further improve dependability and reduce the reaction time in case of failure of the grandmaster clock, the IEEE 1588-2008 standard defines two optional features, namely the grandmaster clusters (subclause 17.3) and the alternate master (subclause 17.4). Besides these static approaches, a solution called master groups was proposed in [42], wherein the reference time is obtained as the average of the times of a group of masters through a fault-tolerant average algorithm. In this way, the failure of one (or more) master(s) only marginally affects the overall accuracy of the group of masters and the synchronized slaves.

DC does not foresee any specific mechanism to increase dependability. In fact, fault tolerance is rarely required on the shop floor, since all the slaves and the network have to be in the working state to allow the correct operation of control applications. This implies that the synchronization mechanism must also be working properly. A certain degree of flexibility is ensured in DC too, as the reference clock is not fixed but selected as the first slave in the segment. Moreover, a double-ring topology can be optionally adopted in EtherCAT (and in SERCOS III as well), which increases the communication reliability. In the case of EPL, the concept of redundant MN is defined in [43] to increase dependability.

Even in this respect, FR is quite different from the other protocols because dependability and fault tolerance are mandatory in automotive systems. Therefore, synchronization is ensured as long as at least a minimum number of nodes are in the operational state, according to a never-give-up strategy.

Conclusions

At present, several solutions are practically available to achieve accurate clock synchronization in distributed control systems, wherein devices are interconnected through a digital communication network. Modern industrial plants, energy distribution, and networked embedded control systems are typical examples of application areas that can benefit from the availability of such a support. Three popular special-purpose synchronization protocols have been considered and analyzed here, i.e., the DC mechanism of EtherCAT, the synchronization technique in EPL, and FR. General-purpose solutions, i.e., NTP and PTP, along with some related implementations in commercial industrial networks (i.e., EtherNet/IP and PROFINET IO) were dealt with in detail in [6].

Although a number of other protocols exist that are able to address this problem, the solutions mentioned previously provide a meaningful picture of the issues typically involved in time synchronization and the techniques used to overcome them. Protocols differ in the type and size of the underlying network, the synchronization approach (centralized versus distributed), and the relationships with communication (ranging from complete independence to tight integration). In turn, these technical choices significantly affect the quality of the attainable synchronization, with accuracies that may range from hundreds of milliseconds down to tens of nanoseconds and even less. Flexibility and dependability are other important issues related to the use of synchronization protocols in real-time distributed control applications.

Security aspects and synchronization over wireless networks, though of primary relevance, were not considered in this article. This omission was intentional because a proper discussion of these topics would have enlarged the scope of the article too much.

Biographies

Gianluca Cena (gianluca.cena@ieiit.cnr.it) has been director of research at the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy, of the National Research Council of Italy, since 2005. His research interests include industrial communication systems and, particularly, real-time protocols. He is a Senior Member of the IEEE. His recent research activities have also included accurate distributed synchronization mechanisms for both wired and wireless networks. He served as program cochair for the 2006 and 2008 editions of the IEEE Workshop on Factory Communication Systems. He has been associate editor of *IEEE Transactions on Industrial Informatics* since 2009.

Ivan Cibrario Bertolotti (ivan.cibrario@ieiit.cnr.it) has been a researcher with the National Research Council of Italy since 1996. Currently, he is with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy. He is a Member of the IEEE. 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. He has coauthored a book on real-time, embedded operating systems and regularly serves as a technical referee for several primary international conferences and journals.

Stefano Scanzio (stefano.scanzio@ieiit.cnr.it) received his Laurea and Ph.D. degrees in computer science from the Politecnico di Torino, Italy, in 2004 and 2008, respectively. Since 2009, he has been working with the National Research Council of Italy. Currently, he is with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy. He is a Member of the IEEE. His current research interests include communication protocols, industrial communication systems, real-time networks, and real-time operating systems. He has gained expertise in synchronization algorithms by developing, studying, and evaluating the accuracy of some of the most popular synchronization protocols.

Adriano Valenzano (adriano.valenzano@ieiit.cnr.it) is director of

research of the National Research Council of Italy. He is currently with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy, where he is responsible for research concerning industrial computer networks and systems. He is a Senior Member of the IEEE and vice president of the Piedmont Chapter of the Italian National Association for Automation. In more than 30 years of scientific activities, he has been involved in many research projects concerning distributed and networked industrial applications with real-time constraints. He has coauthored about 200 refereed journal and conference papers in the area of computer engineering. Since 2007, he has served as an associate editor for *IEEE Transactions on Industrial Informatics*.

Claudio Zunino (claudio.zunino@ieiit.cnr.it) received his degree in computer engineering and his Ph.D. degree in software engineering from the Politecnico di Torino, Italy, in 2000 and 2005, respectively. Since 2006, he has been a researcher with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy, of the National Research Council of Italy. His research interests include wireless communication, industrial Ethernet protocols, computer graphics, parallel and distributed computing, and scientific visualization. He serves as a reviewer for a number of international conferences and journals.

References

[1] *Industrial Communication Networks—Profiles—Part 1: Fieldbus Profiles—CP 3/1 (PROFIBUS DP) and CP 3/2 (PROFIBUS PA)*, IEC 61784-1, Ed. 3.0, 2010.

[2] *CANopen Application Layer and Communication Profile*, CiA 301, Ver. 4.2.0, 2011.

[3] D. L. Mills, "A brief history of NTP time: Confessions of an Internet timekeeper," *ACM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 9–22, Apr. 2003.

[4] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE 1588-2008 (Revision of IEEE 1588-2002), 2008.

[5] T. Sauter, "The continuing evolution of integration in manufacturing automation," *IEEE Ind. Electron. Mag.*, vol. 1, no. 1, pp. 10–19, 2007.

[6] G. Cena, I. Cibrario Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Synchronize your watches. General-purpose solutions for distributed real-time control," *IEEE Ind. Electron. Mag.*, vol. 7, no. 1, pp. 18-29, Mar. 2013.

[7] P. Loschmidt, R. Exel, and G. Gaderer, "Highly accurate timestamping for Ethernet-based clock synchronization," *J. Comput. Netw. Commun.*, vol. 2012, pp. 1–11, 2012.

[8] *Timing Characteristics of Synchronous Ethernet Equipment Slave Clock (EEC)*, ITU-T G.8262, 2007.

[9] M. Lipinski, T. Wlostowski, J. Serrano, and P. Alvarez, "White rabbit: A PTP application for robust sub-nanosecond synchronization," in *Proc. Int. IEEE Symp. Precision Clock Synchronization Measurement Control and Communication (ISPCS)*, Sept. 2011, pp. 25–30.

[10] *Communication Networks and Systems in Substation*, IEC 61850, 2003.

[11] C. De Dominicis, P. Ferrari, A. Flammini, S. Rinaldi, and M. Quarantelli, "On the use of IEEE 1588 in existing IEC 61850-based SASs: Current behavior and future challenges," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 9, pp. 3070–3081, Sept. 2011.

[12] S. Shaheen, D. Heffernan, and G. Leen, "A comparison of emerging time-triggered protocols for automotive X-by-wire control networks," *Proc. Inst. Mech. Eng. Part D J. Automobile Eng.*, vol. 217, no. 1, pp. 13–22, Jan. 2003.

[13] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1204–1223, June 2005.

[14] N. Navet and F. Simonot-Lion. (2009). A review of embedded automotive protocols. *Automotive Embedded Systems Handbook* (Industrial Information Technology Series) [Online], pp. 4.1–4.31. Available: http://hal.inria.fr/inria-00336168

[15] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3-CP 2/2 (EtherNet/IP) and 2/2.1 (EtherNet/IP and Time Synchronization)*, IEC 61784-2, Ed. 2.0, 2010.

[16] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3-CP 3/4-5-6 (PROFINET IO Class A-B-C)*, IEC 61784-2, Ed. 2.0, 2010.

[17] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3-CP 13/1 (EPL)*, IEC 61784-2, Ed. 2.0, 2010.

[18] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3-CP 12/1 (EtherCAT Simple IO) and CP 12/2 (EtherCAT Mailbox and Time Synchronization)*, IEC 61784-2, Ed. 2.0, 2010.

[19] *Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3-CP 16/3 (SERCOS III)*, IEC 61784-2, Ed. 2.0, 2010.

[20] Road vehicles—FlexRay communications system—Part 2: Data link layer specification, ISO 17458-2, Ed. 1, 2013.

[21] Society of Automotive Engineers (SAE). (2011, Nov.). Time-triggered Ethernet. SAE International. [Online]. Available: http://standards.sae.org/as6802

[22] S. Soucek, T. Sauter, and T. Rauscher, "A scheme to determine QoS requirements for control network data over IP," in *Proc. 27th Annu. Conf. IEEE Industrial Electronics Society (IECON)*, 2001, vol. 1, pp. 153–158.

[23] S. Soucek, T. Sauter, and G. Koller, "Effect of delay jitter on quality of control in EIA-852-based networks," in *Proc. 29th Annu. Conf. IEEE Industrial Electronics Society (IECON)*, Nov. 2003, vol. 2, pp. 1431–1436.

[24] P. Martí, A. Camacho, M. Velasco, P. Marés, and J. Fuertes, "Synchronizing sampling and actuation in the absence of global time in networked control systems," in *Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Sept. 2010, pp. 1–8.

[25] Beckhoff Automation GmbH. (2010). Hardware Data Sheet ET1100—EtherCAT Slave Controller, Ver. 1.8 [Online]. Available: http://www.beckhoff.com/

[26] Beckhoff Automation GmbH. (2011). EL6692 EtherCAT Bridge Terminal [Online]. Available: http://www.beckhoff.com/

[27] Beckhoff Automation GmbH. (2010). EL6688 IEEE 1588 external synchronization interface [Online]. Available: http://www.beckhoff.com/

[28] G. Cena, I. Cibrario Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 20–29, Feb. 2012.

[29] G. Cena, I. Cibrario Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "On the accuracy of the distributed clock mechanism in EtherCAT," in *Proc. IEEE Int. Workshop Factory Communication Systems (WFCS)*, May 2010, pp. 43–52.

[30] M. Rehnman and T. Gentzell, "Synchronization in a force measurement system using EtherCAT," in *Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation (ETFA)*, Sept. 2008, pp. 1023–1030.

[31] G. Prytz and J. Skaalvik, "Redundant and synchronized EtherCAT network," in *Proc. Int. Symp. Industrial Embedded Systems, SIES 2010*, pp. 201–204.

[32] *Ethernet POWERLINK, Communication Profile Specification*, EPSG Draft Standard 301, Ver. 1.1.0, 2008.

[33] Ethernet POWERLINK Standardization Group (2013). Industrial Ethernet facts, system comparison, the 5 major technologies [Online]. Available: http://www.ethernet-powerlink.org/

[34] M. Peller, J. Berwanger, and R. Griessbach, "Byteflight specification (draft)—version 0.5, BMW AG," Spec., Oct. 1999.

[35] TTTech. (2003, Nov.). Time-triggered protocol TTP/C high-level specification document—version 1.1, Spec. ed. 1.4.3. [Online]. Available: http://www.ttagroup.org/ttp/specification.htm

[36] R. Froschauer and F. Auinger, "A survey on the integration of the FlexRay bus in distributed automation and control systems," in *Proc. 2nd Int. Symp. Logistics and Industrial Informatics, LINDI 2009*, pp. 1–6.

[37] J. Ungermann. (2009, Dec.). On clock precision of FlexRay communication systems [Online]. Available: http://www.flexray.com/

[38] E. Armengaud, A. Steininger, and M. Horauer, "Towards a systematic test for embedded automotive communication systems," *IEEE Trans. Ind. Informat.*, vol. 4, no. 3, pp. 146–155, Aug. 2008.

[39] 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.

[40] K. Correll and N. Barendt, "Design considerations for software only implementations of the IEEE 1588 precision time protocol," in *Proc. Conf. IEEE 1588 Standard Precision Clock Synchronization Protocol Networked Measurement and Control Systems*, 2006, pp. 1–6

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

[42] 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.

[43] *Ethernet POWERLINK, Part A: High Availability*, EPSG Draft Standard 302-A, Ver. 1.0.0., 2008