Priority-driven Swapping-based Scheduling of Aperiodic Real-Time Messages over EtherCAT Networks

Lucia Lo Bello, *Senior Member, IEEE,* Enrico Bini, *Senior Member, IEEE,* and Gaetano Patti, *Member, IEEE*

***Abstract*—Real-Time Ethernet (RTE) technologies are becoming increasingly popular, as they provide high bandwidth and are able to meet the requirements of industrial real-time communications. Among RTE protocols, the EtherCAT standard is suitable for motion control and closed-loop control applications, which require very short cycle times. As EtherCAT was specifically devised for periodic traffic, aperiodic real-time transmissions are far from being efficient, as they entail long cycle times. To overcome this limitation, this paper presents a general framework for priority-driven swapping-based scheduling of aperiodic real-time messages over EtherCAT networks, which uniformly covers both dynamic and static priority and allows for very short cycle times. The paper provides a description of the priority-driven swapping framework, a schedulability analysis for both static priority and dynamic priority scheduling, and simulative assessments, obtained through OMNeT++ simulations.**

***Index Terms*—Industrial networks, Real-time Ethernet, EtherCAT, Priority-based real-time scheduling.**

I. Introduction and Motivation

THE integration of heterogeneous applications with different information flows and requirements requires networks capable to support multi-service communications. In particular, modern industrial networks must offer support for both time-driven and event-driven control applications. In time-driven applications, messages are periodically transmitted and control actions are taken at constant rate [1], [2], while in event-driven applications, messages are transmitted when one or more trigger events occur (e.g., if the controlled variable exceeds a given threshold) [3], [4]. For example, closed-loop control applications typically generate periodic messages with deadlines of approximately 1 ms [2]. However, these applications may also require the transmission of aperiodic real-time messages, that have to be accommodated in the overall traffic schedule without affecting periodic messages. Aperiodic real-time transmissions are also found in Cyber-Physical Systems (CPSs), as they typically operate in unpredictable environments [5]–[7].

Recently, Real-Time Ethernet (RTE) technologies [8] have become increasingly popular, as they offer high bandwidth, meet the requirements of industrial real-time communication and allow for vertical integration of the different levels in the automation pyramid [9]. Recent literature highlighted the properties of industrial Ethernet networks able to support various traffic classes [10] and temporal constraints [11]. One example is Profinet IRT, whose bandwidth management and scheduling are addressed in [12] and [13], respectively. Another interesting real-time Ethernet protocol is TTEthernet [14], which offers three different traffic classes to support the temporal and bandwidth constraints of a broad range of applications.

This paper focuses on the EtherCAT protocol, which is included in both the IEC 61158 [15] and IEC 61784 [8] standards. EtherCAT is suitable for motion control and closed-loop control applications, which require very short cycle times, where the cycle time is defined as the time necessary to exchange the input/output data between the controller and all the networked devices once [16], [17].

EtherCAT provides a daisy-chain topology and a master/slave architecture in which the master periodically transmits a standard Ethernet frame that embeds an EtherCAT frame containing multiple *telegrams* (as shown in Fig.1). Slaves read and/or write data in the telegram by processing the frame "on-the-fly", so when a byte arrives to a slave it is processed and transmitted to the next slave without waiting for the complete reception of the Ethernet frame. The last slave in the chain transmits the frame back to the master by exploiting the full-duplex capability of Ethernet.

As shown in Fig.1, the EtherCAT frame used for transmitting process data contains one or multiple telegrams, which start with a header containing the command code (i.e., read, write or read/write), the addressing fields, and the payload length. Telegrams end with a Working Counter (WKC) field, which is incremented by the slaves every time they successfully read and/or write data into the telegram. The WKC is also used for error detection.

In order to allow slaves to transmit periodic real-time traffic, the EtherCAT standard provides polling-based mechanisms that can also be adopted for the transmission of aperiodic real-time messages. For instance, the master may send EtherCAT frames containing at least one telegram for each slave that might have aperiodic real-time traffic to transmit.

However, such a mechanism would entail long cycle times, as the master must provide room in the EtherCAT frame for any slave that has the potential to transmit aperiodic real-time traffic, regardless of whether such a slave actually has traffic

Manuscript received February 15, 2014. Accepted for publication August 8, 2014.

Lucia Lo Bello and Gaetano Patti are with the Department of Electrical, Electronics and Computer Engineering, University of Catania, 95125, Catania, Italy (e-mail: lucia.lobello@dieei.unict.it, gaetano.patti@dieei.unict.it)

Enrico Bini is with the Scuola Superiore Sant'Anna, 56124, Pisa, Italy (e-mail: enrico.bini@sssup.it).

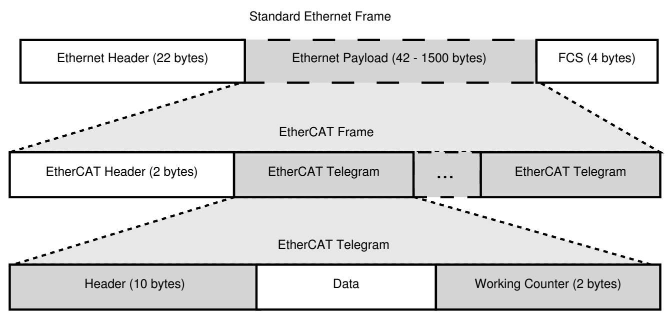

Fig. 1. Structure of an EtherCAT frame containing multiple telegrams

to transmit. For instance, in a network with 20 slaves, each with the potential for transmitting 32 byte long aperiodic real-time messages, the master should provide 20 telegrams for each cycle. This would result in an increase in the cycle time duration of 70.4 μs, with the probability that most of the time such telegrams would not actually be used, due to the event-driven nature of aperiodic traffic generation.

In addition to the cycle time increase, the need for handling aperiodic real-time messages by a polling mechanism performed by the master would also introduce, in the case of event-triggered control applications, a bandwidth waste that would reduce the main advantage of the event-triggered control, that is, the low bandwidth demand.

Some works in the recent literature address methods to reduce the cycle times in the EtherCAT networks. In [18] periodic and aperiodic real-time traffic is scheduled by the master, which provides guaranteed bandwidth for the slave transmissions. In [19] a switch operating at the EtherCAT telegram level is proposed, but such a solution does not focus on aperiodic traffic scheduling.

The work in [20] proposes an arbitration-based access scheme for aperiodic real-time messages which introduces new aperiodic telegrams that are contented by the slaves for transmitting aperiodic messages. The slave with the highest priority among the ones competing for the aperiodic telegrams will overwrite "on-the-fly" the incoming aperiodic message.

The mechanism in [20] foresees that the master transmits a copy of the aperiodic message received. In this way, the slave that transmitted the message realizes that it was successful and so the message can be safely removed from its queue. Conversely, the other slaves know that they did not succeed and must try again. This approach reduces the cycle time for transmitting aperiodic messages compared with the EtherCAT standard, but still suffers from some limitations. The main one is that low priority messages may experience long delays due to the interference from high priority ones, with a potential for starvation.

The same authors in [21] added the capability for embedding multiple aperiodic messages in one telegram. In this new approach, a slave has to receive a cumulative "acknowledgement" message from the master before removing the transmitted message from the local queue. The purpose of such a message is twofold. It is a notification of successful transmission and also a way to allow a slave to remove the aperiodic message from its local queue.

Inspired by the Slot Swapping Protocol (SSP) [22]–[24], in [25] dynamic priorities are exploited to swap between an incoming lower priority aperiodic message and a higher priority message pending at the current slave. This is achieved by defining a new telegram type, contented among the slaves that have an aperiodic real-time message to transmit.

Contention is ruled by comparing the absolute deadlines of the messages according to the EDF algorithm. When the message contained in the incoming telegram is less urgent and therefore loses the contention, it is swapped and replaced by the message with the most urgent deadline in the local queue. Swapping does not entail message loss, because the slave which has swapped the incoming message will be in charge of transmitting it whenever possible.

a) Contributions of this paper: This paper proposes a Priority-driven swapping-based mechanism to deal with the problem of providing support for aperiodic real-time traffic over EtherCAT networks. As previously explained, the EtherCAT standard does not offer this support. The Priority-driven swapping-based approach combines the arbitration mechanism in [21] with the deadline-driven swapping proposed in [25]. In particular, this paper extends the work in [25], which only addresses EDF scheduling, in a general framework for priority-driven swapping-based scheduling of aperiodic real-time messages over EtherCAT networks, which uniformly covers both dynamic and static priority. The proposed mechanism allows slaves to transmit multiple aperiodic real-time messages in a single EtherCAT frame, while maintaining compatibility with the EtherCAT standard and achieving cycle times in the order of $100 \mu s$.

The paper target is to achieve short cycle times while ensuring the schedulability of aperiodic real-time messages. For this reason, the paper provides schedulability conditions that enable the network designer to configure the network parameters (e.g., the number of aperiodic telegrams in the EtherCAT frame) so as to avoid deadline miss. The Priority-driven Swapping here proposed is implementable with minor modifications in the EtherCAT protocol state machine and maintains compatibility with EtherCAT standard devices, so there is a clear potential for industrial exploitation.

The paper is organized as follows. Section II describes the Priority-driven Swapping approach here proposed and discusses its implementation. Sect. III presents the timing analysis of the approach, under static and dynamic priorities, respectively. Sect. IV deals with simulative assessments of the Priority-driven Swapping approach and discusses its performance. Finally, Sect. V concludes the paper and gives hints for future work.

II. The Priority-driven Swapping

In the Priority-driven Swapping approach here proposed to efficiently support aperiodic traffic under both static and dynamic priority scheduling, a novel telegram has been introduced. The aperiodic telegram, which is shown in Fig. 2, is contented among the slaves according to a preemptive policy that enables the slaves to send aperiodic messages when needed. This is possible as the daisy-chain topology of

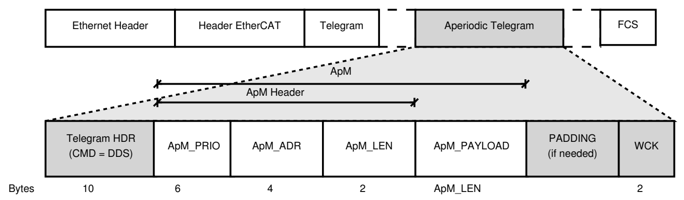

Fig. 2. Aperiodic Telegram structure

TABLE I
APERIODIC TELEGRAM FIELDS.

Data Field	Data Type	Description/Value
CMD	Unsigned8	Command: (PdS) 0x10
IDX	Unsigned8	Index
ADR	DWORD	Slave address of the last ApM
LEN	Unsigned11	DATA field Length
RESERVED	Unsigned3	0x00
C	Unsigned1	Circulating frame
NEXT	Unsigned1	0 if the last telegram in the frame
IRQ	WORD	Reserved for future use
DATA	OctectString	Data
WKC	WORD	Working Counter

EtherCAT allows for message preemption by changing "on-the-fly" the telegram payload of an incoming frame when the frames traverses a slave.

Each slave maintains a local queue of aperiodic messages (ApMs), ordered according to a priority. The network can work under either static or dynamic priority scheduling, but the choice has to be made during the configuration phase, through a suitable setting.

In order to maintain compatibility with the EtherCAT standard, the aperiodic telegram is mapped into the standard EtherCAT telegram as shown in Table I.

The CMD field of the telegram header contains the 0x10 value which indicates an aperiodic telegram, while the ADR field contains the address of the slave that generated the last received ApM. The other fields remain the same as specified in the EtherCAT standard. The DATA field contains an ApM, which is the aperiodic message containing the data transmitted by a slave. The ApM is composed of an ApM header and a payload. The header fields are specified as follows:

- *ApM_PRIO*: the message priority.
- *ApM_ADR*: the address of the slave that transmitted the ApM.
- *ApM_LEN*: the ApM payload length in bytes.

The master periodically transmits one EtherCAT frame containing both standard telegrams for the periodic data and one or multiple aperiodic telegrams. A slave, with a ready-to-transmit ApM, upon receiving the aperiodic telegram, stores the local ApM with the highest priority (M_{loc}) in the output buffer (Out_Buf) and the incoming ApM (M_{in}) in the input buffer (In_Buf). When the first byte of M_{in} arrives, the contention starts and the slave compares byte by byte the priorities of M_{loc} and M_{in}. The message with the highest priority value is transmitted, while the other one is stored in the slave local queue.

For priority comparison, due to serial communication, the six bytes of the ApM_PRIO field are encoded and transmitted from the most to the least significant byte. The comparison works as follows. The i-th byte of the priority of the incoming message M_{in} (i.e., the ApM_PRIO field of the ApM), $B_{i,in}$, for i=0...5, is compared with the corresponding byte of the ApM_DL field of the ApM of the local message M_{loc}, $B_{i,loc}$. The incoming message M_{in} is swapped if the following inequality (1) holds (here we assume that the lower the value, the higher the priority):

$$B_{i,loc} < B_{i,in} \tag{1}$$

otherwise M_{in} is forwarded to the next slave. No swap occurs if the incoming ApM has the same priority as the local ApM. If according to inequality (1) a swap occurs, while the local message M_{loc} is transmitted to the next slave and removed from the local queue, the swapped message has to be entirely received and is then inserted in the local queue according to its priority. Such a mechanism cannot be implemented in the upper layer of the Data Link, as it requires "on-the-fly" processing of the frame.

Compared to the EtherCAT standard [15], the Priority-driven swapping provides the possibility for slaves to transmit aperiodic messages without the need for the master to provide room for each slave with a potential for transmitting, thus reducing the cycle time. In fact, a single aperiodic telegram can be contented among all the slaves that intend to transmit an aperiodic message. If compared to the mechanism proposed in [21], the Priority-driven swapping provides two advantages. First, it allows a slave to embed more aperiodic telegrams into a single EtherCAT frame. This is possible because the slave which has swapped the incoming message will be in charge of transmitting it whenever possible. Second, thanks to this mechanism, a slave which transmitted an ApM can remove it from the local queue right after completing the transmission without waiting for an "acknowledgement" message, as no aperiodic message will be lost due to preemption from other messages.

This is not the case for the CAN-like approach in [21], where a slave must wait for the acknowledgement message sent by the master before removing the transmitted message from the local queue, so the same slave cannot send more than one aperiodic message in the same telegram. The second advantage compared to [21], which only addresses static priorities in a CAN-like fashion, is that the Priority-driven swapping uniformly covers both dynamic and static priority scheduling, therefore, at the configuration step, the most appropriate scheduling policy for the application at hand can be chosen on a case-by-case basis.

An example of dynamic priority scheduling is the Earliest Deadline First (EDF) algorithm [26], in which the messages with closer absolute deadlines preempt those with less imminent ones. To implement EDF in the Priority-driven swapping approach here proposed, the absolute deadline counts the microseconds elapsed since January 1, 2000 (the date refers to the EtherCAT system time [27]) and a slave generating an ApM calculates the absolute deadline by adding to the

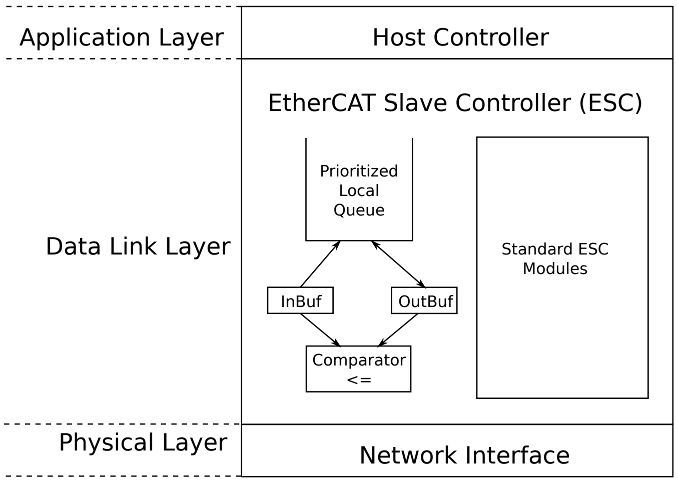

Fig. 3. Modules implementing the Priority-driven Swapping

system time the relative deadline of the message received from the upper layers. Such a mechanism relies on the clock synchronization, which is also provided with the EtherCAT standard with an accuracy in the order of $100ns$ [27].

A. Implementing the Priority-driven Swapping

Each EtherCAT slave consists of three layers, i.e.:

- Application layer, which contains the Host Controller. This can be implemented, for instance, in a microcontroller.
- Data Link Layer, which contains the EtherCAT Slave Controller (ESC). This can be implemented either in hardware (FPGA, ASIC) or in software [28].
- Physical Layer, which implements the physical interface to the network.

The Priority-driven Swapping approach requires an additional module in the ESC whose main components, shown in Fig. 3, are a prioritized local queue containing the local ApMs, two buffers for the incoming and outcoming ApMs, and an 8-bit comparator. All these components can be implemented in hardware in the case of FPGA/ASIC ESC. The hardware implementation of the buffers and the comparator is simple and, as far as the prioritized local queue is concerned, in the literature several hardware implementations of prioritized local queues are proposed [29] (e.g., Shift register prioritized queue or Systolic array prioritized queue). The module can be also implemented in software in the case of software ESC, as shown in [28]. Some slight modification in the EtherCAT protocol state machine has also to be added in order to handle the novel telegram type.

III. Analytic Assessment

A. Frame propagation and Timing

In EtherCAT networks, if the data to be embedded in the Ethernet frame exceed the maximum payload size (i.e., 1500 bytes), multiple Ethernet frames will have to be transmitted by the master to complete a cycle. However, this paper addresses the case of very short cycle times (e.g., in the order of $100\mu s$), and a maximum payload Ethernet frame has a cycle time of $150\mu s$. Hence, here only a single frame cycle is considered.

TABLE II
Summary of notation

Symbol	Definition
m	The number of slaves in the chain
ℓ_k	Length of the cable connecting the k-th slave to the $(k+1)$-th slave. ℓ_0 is the length of the cable connecting the master to slave 1, while ℓ_m the length of the cable connecting slave m back to the master.
P	The transmission period of the Ethernet frame.
T_c	The minimum cycle time, i.e., the minimum time taken by all the network nodes to exchange their input/output data once. In Fig.4, $T_c = T_{et} + T_{ec} + T_{pr} + T_{de} + T_{if}$.
T_{et}	The sum of the transmission times of the Ethernet header and Frame Check Sequence (FCS) fields ($\mathsf{a} + \mathsf{c}$ in Fig. 4).
T_{ec}	The time necessary to transmit the EtherCAT frame.
u	The propagation delay per unit of length over the medium (that is 5 ns/m according to the EtherCAT standard).
T_{pr}	The propagation delay over the communication medium that is equal to $T_{pr} = u\sum_{k=0}^{m} \ell_k$.
T_{sv}	the time taken by each slave to process the frame.
Δ_k	the delay from the reception of a frame at slave k to the reception of the same frame at the master, that is $(m-k+1)T_{sv} + u\sum_{i=k}^{m} \ell_i$.
T_{de}	The frame delay that is mT_{sv}.
T_{if}	The inter-frame gap, i.e., the time between the end of the transmission of an Ethernet frame and the start of the transmission of the next one.
p	Number of aperiodic telegrams in the frame ($p = 3$ in Fig. 4).
T_{ap}	The time between the start of the transmission of the Ethernet frame and the start of the transmission of the first aperiodic telegram ($\mathsf{a} + \mathsf{b} - p\mathsf{q}$ in Fig. 4).
n	The number of ApMs.
π_i	The index of the slave where the i-th ApM is generated.
$\mathsf{pri}(i)$	Priority of the i-th ApM.
T_i	The minimum interarrival time between two consecutive instances of the i-th ApM.
D_i	The relative deadline of the i-th ApM.
S	The time for receiving an aperiodic telegram (q in Fig. 4).
A	Time elapsing from the start of the reception of the first aperiodic telegram to the end of the reception of the frame ($pS + \mathsf{c}$ in Fig. 4).

Fig. 4 illustrates the propagation of the Ethernet frame and the related terminology and notations (also summarized in Table II). The Figure shows a scenario with an Ethernet frame that is transmitted by the master to a chain of three slaves, and then goes back to the master according to the daisy chain topology. In Fig. 4 the master is represented twice to illustrate separately the transmission (top-side) and the reception (bottom-side). The propagation of the Ethernet header and the *Frame Check Sequence* (FCS) field is drawn in light gray.

The Ethernet payload is drawn in two shades of gray (gray/dark gray). In the payload, we highlight in dark gray the EtherCAT telegrams (three in the figure) dedicated to the transmission of aperiodic messages (ApMs). All the notations are described in Table II. The master periodically sends an

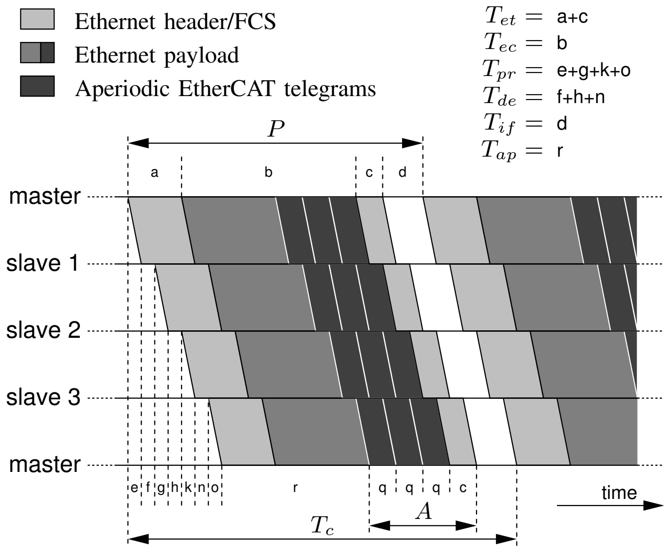

Fig. 4. EtherCAT frame processing sequence

Ethernet frame, with period P. Each frame then propagates to the slaves through the network. Slaves are labeled following the frame reception order, so slave 1 is the one that receives the frame first, while slave m is the last. The time elapsing from the transmission of a frame by the master to its reception, as a response [15], is equal to the time needed by the signal to propagate through the medium (T_{pr}), plus the time needed by the m slaves to process the frame (T_{de}).

From Fig. 4 it is possible to observe that the slaves process the frame "on-the-fly", i.e., each frame starts to be transmitted before it has been fully received from the preceding node. This allows a low end-to-end latency and enables the transmission of an ApM even if the external event generating the ApM arrives during the reception of the Ethernet frame. At any slave, the instant at which the highest priority ApM is transmitted is the start of the reception of the next aperiodic telegram (here called start time). Then, the availability of aperiodic telegrams must be carefully analyzed, as illustrated in the following.

B. Timing Analysis

In real-time systems, in which the focus is on meeting the deadlines in the worst case, the analysis is made by determining the worst case for both the resource availability and the resource request [30]. In the analysis the same number of aperiodic telegrams for each EtherCAT frame is assumed.

In this context, the resource is the start time of an aperiodic telegram. Hence, the worst case for the resource availability is determined by computing the minimum number $s(t)$ of start times of aperiodic telegrams which may occur in any interval of length t. As illustrated in Fig. 5, the worst-case interval (that is the one containing the minimum number of start times of aperiodic telegrams) begins when the last aperiodic telegram has just started. From this instant (please refer to Fig. 5 where the black dots denote the start time of the aperiodic telegrams over time), the time a slave may have to wait before another aperiodic telegram starts is $P-(p-1)S$. Then p aperiodic telegrams will start, spaced S, and the pattern will repeat with period P.

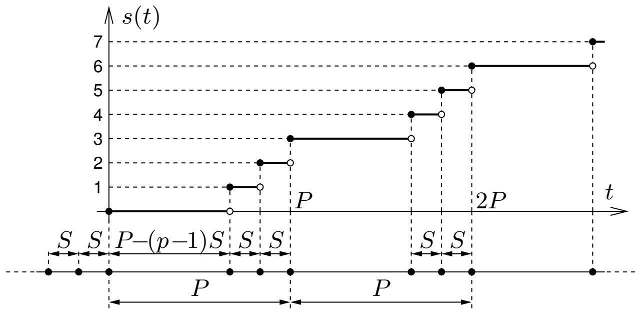

Fig. 5. Example of the minimum number of start times $s(t)$, with $p = 3$ aperiodic telegrams in a frame.

The formal expression of $s(t)$ can then be written as follows

$$s(t) = \sum_{j=1}^{p} \left\lfloor \frac{t+(j-1)S}{P} \right\rfloor, \tag{2}$$

which accounts for the fact that:

- the start times of the p aperiodic telegrams in the same frame are separated by S;
- consecutive aperiodic telegrams at the same position in the frame are separated by P.

In the example shown in Figure 5, we have $p = 3$ aperiodic telegrams in a frame.

C. Response-time analysis

For computing the longest response time of an ApM, we adopt the classic busy-interval approach [31], which was extended to the event-stream task model [32] by Ritcher et al [33]. To apply this method, it is necessary to compute the longest time $w(N)$ a slave has to wait to see N consecutive start times of aperiodic messages. By this definition, $w(N)$ is given by

$$w(N) = \sup\{x \in \mathbb{R} : s(x) < N\}. \tag{3}$$

By definition, $w(N)$ then is the longest time interval in which less than N aperiodic telegrams may start.

In the special case of $s(t)$ of (2), consecutive aperiodic telegrams in the same frame are separated by S, while the last aperiodic telegram in the frame and the first one in the next frame are separated by $P-(p-1)S$, so that aperiodic telegrams at the same position in the frame are separated exactly by P. As also illustrated in Figure 5, this condition implies that $w(N)$ increases by $P-(p-1)S$ when N is a multiple of p, while it increases by S at all other values of N (not multiple of p). Hence $w(N)$ can be written as

$$w(N) = (Q+1)P - (p-1-Z)S \tag{4}$$

with Q and Z, respectively, quotient and remainder of the Euclidean division of $N-1$ by p, that is $N-1 = Qp+Z$ with $Z \in \{0,1,\ldots,p-1\}$. For example, if $p = 3$ (as illustrated in Fig. 5) and $N = 1$, then the result of the Euclidean division is $Q = 0$ and $Z = 0$. For this choice, from (4) we find $w(1) = P-2S$, which correctly is the longest separation between two consecutive aperiodic telegrams. If, for example, $N = 3$, then $Q = 0, Z = 2$ and $w(3) = P$. With these definitions in mind, we can compute the response time of an ApM, by applying the classical busy-period approach. First, we define $I_j(t)$ as the largest number of messages of the j-th ApM which can be generated in any interval of length t. For example, in the classical example of sporadic ApM with minimum interarrival time T_j, we have $I_j(t) = \left\lceil \frac{t}{T_j} \right\rceil$. Then, we compute the largest number N_i of aperiodic telegrams, which may be needed to transmit the i-th ApM, as the smallest fixed point of the following iteration

$$\begin{cases} N_i^{(0)} & = 1 \\ N_i^{(k+1)} & = 1 + \sum_{\mathsf{pri}(j)>\mathsf{pri}(i)} I_j(w(N_i^{(k)})) \\ & \quad + \sum_{\substack{\mathsf{pri}(j)=\mathsf{pri}(i) \\ \pi_j<\pi_i}} I_j(w(N_i^{(k)})) \\ & \quad + \sum_{\substack{\mathsf{pri}(j)=\mathsf{pri}(i) \\ \pi_j=\pi_i, j\neq i}} 1. \end{cases} \tag{5}$$

The expression of $N_i^{(k+1)}$ accounts for the fact that in the time interval $w(N_i^{(k)})$ we must consider the following possible sources of interference:

- the ApMs with higher priority, represented by the first sum with j such that $\mathsf{pri}(j) > \mathsf{pri}(i)$;
- the ApMs with the same priority as the i-th, but generated at preceding slaves (as described earlier, an incoming ApM with the same priority of the ApM at the local slave is not swapped), represented by the sum over $\mathsf{pri}(j) = \mathsf{pri}(i), \pi_j < \pi_i$;
- the ApMs at the same slave with the same priority, which, however, may interfere only once, since ApMs with the same priority are scheduled FIFO within the same slave (the last term in the sum of (5)).

Note that, if messages have minimum interarrival time T_j, the above described iterative definition is proved to converge [31] if the maximum number of needed aperiodic telegrams is less than the available ones, that is

$$\sum_{\mathsf{pri}(j)>\mathsf{pri}(i)} \frac{1}{T_j} < \frac{p}{P}. \tag{6}$$

Once N_i is computed, the response time (R_i) of the i-th ApM is

$$R_i = \Delta_{\pi_i} + w(N_i) + A \tag{7}$$

with the following interpretation:

- Δ_{π_i} is the maximum time from the slave π_i to the master;
- $w(N_i)$ is the time the i-th ApM may have to wait to have one telegram available;
- A is the time from the reception of the aperiodic telegram at the master to the instant the message is actually read.

D. EDF analysis

If the ApMs have a deadline D_i between their release at the slave and the delivery time at the master, and ApM priorities are assigned based on EDF, so it is then possible to tighten the analysis by adapting the classic EDF guarantee test [34].

Theorem 1: A set of n aperiodic messages (ApMs), each one triggered by events with minimum interarrival time T_i are guaranteed to be received at the master within D_i from their release time when scheduled by EDF, if:

$$\forall t > 0, \quad \sum_{i=1}^{n} \max\left(0, \left\lfloor \frac{t - (D_i - \Delta_{\pi_i} - A)}{T_i} \right\rfloor + 1\right) \leq s(t). \tag{8}$$

Condition (8) also implies the necessary condition

$$\sum_{i=1}^{n} \frac{1}{T_i} \leq \frac{p}{P} \tag{9}$$

which is equivalent to the classic necessary condition of a non-overloaded processor in CPU scheduling problems.

Theorem 1 (whose proof is given in Appendix A) provides a test which is not practical. In the next Lemma (whose proof is given in Appendix B), following the idea by Baruah et al [35], we reduce the set in which the inequality of (8) needs to be tested to a finite one.

Lemma 1: A set of n aperiodic messages (ApMs), each one triggered by events with minimum interarrival time T_i are guaranteed to be received at the master within D_i from their release time when scheduled by EDF, if:

$$\sum_{i=1}^{n} \frac{1}{T_i} < \frac{p}{P} \tag{10}$$

and

$$\forall t \in \mathcal{D}, \quad \sum_{i=1}^{n} \max\left(0, \left\lfloor \frac{t - \phi_i}{T_i} \right\rfloor\right) \leq s(t) \tag{11}$$

with

$$\phi_i = D_i - \Delta_{\pi_i} - A - T_i \tag{12}$$

$$L^* = \max_{\ell=0,\ldots,n} \frac{\frac{p}{P}(P - (p-1)S) - \sum_{i=1}^{\ell} \frac{\phi_i}{T_i}}{\frac{p}{P} - \sum_{i=1}^{\ell} \frac{1}{T_i}} \tag{13}$$

$$\mathcal{D} = \{d : d = \phi_i + kT_i, i = 1, \ldots, n, k \in \mathbb{N}, d < L^*\}, \tag{14}$$

and assuming that ApMs are sorted by an increasing value of ϕ_i.

IV. Simulative assessments

A. Simulation Model Assessment

To assess the performance of the proposed approach, a suitable simulation model was developed using the OMNeT++ framework. In the simulation model two kinds of nodes are implemented, i.e., the EtherCAT Master and the EtherCAT Slave. The EtherCAT Master is composed of a *MasterDLL* module and a *MasterPHY* module. The first module periodically generates Ethernet frames and collects statistics. The MasterPHY transmits the frame in one-byte long packets and transmits each byte every $0.08\mu s$ (i.e., the byte time of the 100Mb/s Ethernet). In this way, the timing of the simulation model is compliant with that of the EtherCAT standard. The EtherCAT Slave module provides several functionalities. Among them, the EtherCAT DLL, which supports both the periodic telegrams foreseen by the standard and the aperiodic telegrams of the proposed Priority-driven swapping approach, the forwarding mechanisms for incoming packets,

TABLE III
SIMULATION I - NETWORK PARAMETERS AND TIMINGS

Net. params	Value/range
T_{sv}	$1\mu s$
T_{pr}	$50ns$
T_c	$46.33\mu s$
P	$41.28\mu s$
A	$3.84\mu s$
p	1
Payload size of an aperiodic telegram	44 bytes
Number of periodic telegrams	7
Payload size of a periodic telegram	48 bytes

TABLE IV
SPORADIC TRAFFIC PARAMETERS OF SIMULATION I

ApMs Set	Slave π_i	T_i	D_i	Δ_k
1	1	$500\mu s$	$500\mu s$	$5.040\mu s$
1	2	$500\mu s$	$500\mu s$	$4.030\mu s$
2	1	$1000\mu s$	$1000\mu s$	$5.040\mu s$
2	2	$1000\mu s$	$1000\mu s$	$4.030\mu s$
2	3	$1000\mu s$	$1000\mu s$	$3.020\mu s$
2	4	$1000\mu s$	$1000\mu s$	$2.010\mu s$
2	5	$1000\mu s$	$1000\mu s$	$1.000\mu s$

the read/write and management functions of the ApMs local queue, and the slave Application layer.

The simulation model was assessed by comparing the EtherCAT timing parameters calculated as in Sect. III with those obtained in the simulation. In particular, for the simulations, a typical networked control system for motion control was simulated. The system consists of one controller (the master), 5 devices (the slaves), 6 joints and 4 wheels. Three slaves are in charge of 2 axes each, and the other two slaves manage 4 wheels (2 wheels for each slave). For the joint control 48-byte data is cyclically exchanged with a period of $50\mu s$, a realistic value for these applications [36]. The wheels are controlled in an event-triggered way, so the traffic is sporadic (i.e., characterized by a minimum interarrival time). The data transmission period for the wheel control is $500\mu s$ [37] and data size at the application layer is 32 bytes. The wheel control traffic is randomly generated between $500\mu s$ and $1000\mu s$ using a uniform distribution. The relative deadline D_i is chosen equal to the minimum interarrival time (i.e., $500\mu s$). The slaves may also transmit event notification messages characterized by $T_i = 1000\mu s$ and $D_i = 1000\mu s$. The distance between two consecutive network nodes is 2m, so the overall distance covered by the Ethernet frames is 20m. The relevant simulation parameters and timing are summarized in Table III.

In the simulation, the slaves 1 and 2 manage the wheels, while the slaves 4,5, and 6 are in charge of the 6 joints. Table IV shows the ApMs parameters. An ApM set indicates the ApMs with the same interarrival time T_i and the same relative deadline D_i, and Δ_k is the delay from the reception of a frame at slave k to the reception of the same frame at the master (as defined in Table II). With these parameters, the conditions of Lemma 1 are met. Therefore, the scheduling of the ApM set is feasible.

A simulation run of 10s, corresponding to a generation of about 20000 ApMs, was performed and simulations were repeated 5 times varying the seed of the random generators. The calculated cycle time is $46.33\mu s$ as required from the application. Results show that the response time values obtained by the simulation match the ones calculated using the analysis described in Section III. In Fig. 6 the Cumulative Percentage Distribution of the message response times, defined as the percentage of ApMs with the response time lower than a given response time value (i.e., the value of x-axis), is drawn.

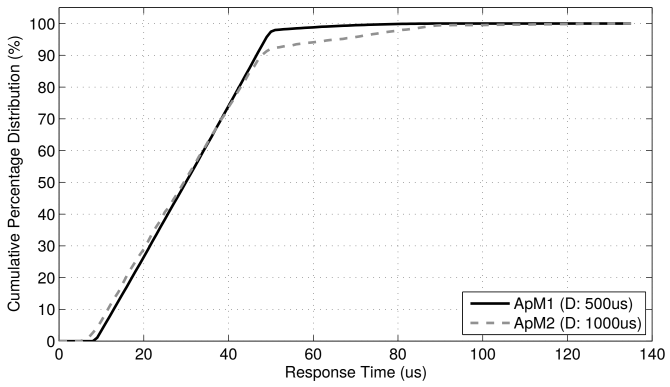

Fig. 6. Cumulative Percentage Distribution of the ApM Response Times.

This corresponds to the results of the analysis that proved the feasibility of the considered ApM set (i.e., that the ApM response times are always lower than their relative deadlines) in the addressed scenario. The same simulation was performed using static priorities. Results also match with the values obtained with the analysis in Sect. III-C.

B. Cycle Time assessment

To assess the cycle times which can be reached using the proposed approach and to compare it with the standard EtherCAT, a set of simulations were performed. The goal of these simulations is to find the minimum number of aperiodic telegrams required to transmit the ApMs while maintaining the Deadline Miss Ratio (DMR) (i.e., the number of deadline misses over the number of generated ApMs) lower than 0.1%.

In this simulation a network with 10 slaves was deployed. Each slave generated ApMs with three different priorities (or deadlines in the case of EDF scheduling). ApMs were generated with exponentially distributed random generation periods with a given mean (λ), while the ApMs deadlines were uniformly distributed. Table V shows the simulation parameters. Lambda is given as an interval because for each simulation run different lambda values were used. This choice was made in order to evaluate the protocol performance with varying mean generation periods.

To have realistic frame sizes, 20 periodic telegrams were transmitted, in addition to the aperiodic telegrams, every cycle. This entails an increase of $70.4\mu s$ in the cycle time. Each

TABLE V
PARAMETERS OF SIMULATION II

Parameters	Values/Range
Number of slaves	10
Number of periodic telegrams	20
Payload size of a periodic telegram	32 bytes
Number of aperiodic telegrams (p)	from 1 to 8
Payload size of an aperiodic telegram	32 bytes
ApMs deadlines	$300\mu s, 600\mu s, 900\mu s$
λ	from $186\mu s$ to $1515\mu s$
Repetitions	5 repetitions varying the seed

TABLE VI
CYCLE TIME AS A FUNCTION OF THE NUMBER OF APERIODIC TELEGRAMS IN THE ETHERCAT FRAME

p	T_c	p	T_c
1	$87.62\mu s$	5	$101.70\mu s$
2	$91.14\mu s$	6	$105.22\mu s$
3	$94.66\mu s$	7	$108.74\mu s$
4	$98.18\mu s$	8	$112.26\mu s$

simulation run was repeated 5 times varying the seed. In each simulation the simulated time was chosen to collect statistics over 50000 ApMs.

Increasing the number of aperiodic telegrams embedded in the EtherCAT frame entails an increase in the cycle time. For the simulated scenario, the cycle time values as a function of the number of aperiodic telegrams are shown in Table VI.

Fig. 7 compares the simulation results obtained by the standard EtherCAT, the Priority-driven Swapping with dynamic priorities (PdS-EDF), and the Priority-driven Swapping with static priorities (PdS-SP).

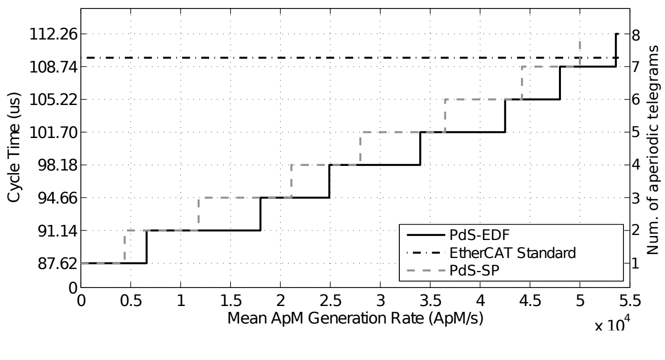

Fig. 7. Cycle Time vs. Mean ApM Generation Rate

In the EtherCAT standard simulation, one periodic telegram with 20 bytes of payload is transmitted by each slave, so the cycle time is constant in Fig. 7 (i.e., $109.70\mu s$). Conversely, in the PdS-EDF and in the PdS-SP the cycle time varies as a function of the aperiodic workload. This is an advantage of the two mechanisms here proposed over the EtherCAT standard. In fact, while the latter provides a constant cycle time, that is higher than 109 us in Fig. 7, independently of the actual aperiodic workload, the two Priority-driven Swapping approaches obtain a cycle time that depends on the aperiodic workload. As a result, under low aperiodic workloads, the cycle time is also low. As shown in Fig. 7, the two Priority-driven Swapping approaches experience a cycle time value comparable with that of the EtherCAT standard only under a high aperiodic workload, i.e., 53700 aperiodic messages per second. The Priority-driven Swapping (PdS) shows the same trend with both static (PdS-SP) and dynamic priorities (PdS-EDF), with the difference that with the same number of aperiodic telegrams embedded in one EtherCAT frame, and so with the same cycle time, the PdS-EDF supports a higher generation rate for aperiodic real-time messages than the PdS-SP. This is because in the PdS-SP the low-priority messages experience longer delays due to the interference from the messages with higher priority. We highlight that in both the Priority-driven Swapping approaches the number of ApMs in the slave queue is always lower than 10 ApMs, so the local queues were never saturated. Summarizing, the Priority-driven Swapping offers better performance than the EtherCAT standard in terms of reduction of the cycle time for the aperiodic traffic.

C. Comparison with the CAN-Like

In this simulation a comparison between the Priority-driven Swapping with another approach proposed in the literature [21], here called ¨*CAN-Like*¨, is performed. In particular, the response times in a defined scenario are compared.

In the CAN-Like an aperiodic telegram, called *MARB*, is embedded in the EtherCAT frame. Slaves transmit aperiodic messages in the MARB using a ¨bytewise¨ arbitration (similar to the CAN protocol). The CAN-Like approach allows the possibility to transmit multiple aperiodic messages in a MARB (but only one for each slave). Moreover, as the CAN-Like does not provide any swapping mechanism, it requires an additional telegram to inform the slaves about the received messages (i.e., the MDBT Telegram). So, if the slave messages are received by the master, then the slaves can remove the pending messages from their local queue. This prevents the master from embedding multiple MARBs in the same EtherCAT frame and also prevents the slaves from transmitting multiple aperiodic messages within a single MARB.

In order to compare the two protocols a network with 10 slaves was simulated. Each slave generates aperiodic messages according to an exponential distribution with mean λ. A simulation was performed using static priorities. The priority of each message was assigned randomly with a uniform distribution between three ranges, as shown in Table VII. As the CAN-Like approach does not provide the possibility to generate messages with the same priorities (as the priority identifies the message), the index of each slave is subtracted to the priority, so different slaves generate messages with different priority (e.g., slave 10 subtracts 10 to the priority of its messages). The same assignment has been used for the Priority-driven Swapping here proposed. In Table VII the parameters of this simulation are shown.

A tuning of the MARB length (for CAN-Like) and of the number of aperiodic telegrams (for PdS) was performed in order to choose the best simulation parameters regarding the

TABLE VII
PARAMETERS OF SIMULATION III

Parameters	**Values/Range**
Number of slaves	10
Number of periodic teleg.	20
Payload size (periodic teleg.)	32 bytes
Number of aperiodic teleg. (p)	4 (PdS), 1 (CAN-Like)
Payload size (aperiodic teleg.)	32 bytes (PdS), 50 bytes (CAN-Like)
Repetitions	5 repetitions varying the seed
λ	$1162\mu s$ (i.e., 8600 messages/s)
Aperiodic message priorities	High:$[600, 609]$ μs, Medium:$[900, 909]$ μs, Low:$[1200, 1209]$ μs

response times. For the CAN-Like a MARB of 50 bytes was set, so that 2 aperiodic messages per cycle can be transmitted (increasing the MARB size does not result in better performance). The mean generation period for the aperiodic messages is $1162\mu s$, which corresponds to a workload of 8600 messages/s, so as not to saturate the network. The Cumulative Percentage Distribution of response times is shown in Fig. 8.

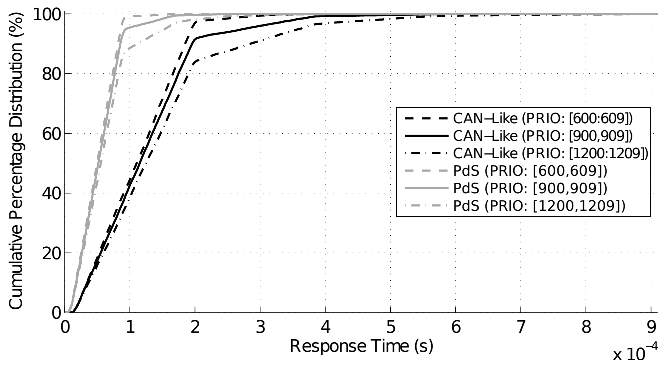

Fig. 8. Cumulative Percentage Distribution of Response Times (CAN-Like vs. PdS)

Taking into account 80% of the aperiodic messages, the Priority-driven Swapping (PdS) obtains response times lower than $100\mu s$, while the response times obtained by the CAN-Like are higher than $100\mu s$, but remain lower than $200\mu s$. Considering the whole set of messages, the maximum response time for the highest priority messages is $214\mu s$ for the PdS and $532\mu s$ for the CAN-Like, while the maximum response time for the lowest priority messages is $406\mu s$ for the PdS and $879\mu s$ for the CAN-Like. The Priority-driven Swapping provides lower response time than the CAN-Like thanks to the capability for the same slave to transmit multiple aperiodic telegrams in the same EtherCAT frame.

V. CONCLUSIONS

In this work a Priority-driven swapping mechanism to deal with the problem of providing support for aperiodic real-time traffic over EtherCAT networks was presented. The proposed mechanism provides the possibility for slaves to transmit aperiodic real-time messages under static and dynamic priorities, respectively, while maintaining the compatibility with the EtherCAT standard. The proposed approach is suitable for event-triggered control applications, as the aperiodic messages can be transmitted while maintaining short cycle times (i.e., in the order of $100\mu s$). The paper proposed a general analysis for the Priority-driven scheduling based on response times which can be used to assess the feasibility of a static priority message set. As far as dynamic priorities are concerned, the case of EDF scheduling was analytically assessed and the relevant schedulability conditions were derived. The paper also provided extensive simulative assessments performed through the OMNeT++ framework. Future works will address the possibility to embed multiple ApMs within a single aperiodic telegram so as to further reduce the cycle time.

REFERENCES

[1] A. Cuenca, J. Salt, A. Salaand, and R. Piza, "A Delay-Dependent Dual-Rate PID Controller Over an Ethernet Network," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, 18-29, Feb. 2011.

[2] K. W. Schmidt and E. G. Schmidt, "Distributed Real-Time Protocols for Industrial Control Systems: Framework and Examples," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 10, 1856-1866, Oct. 2012.

[3] A. Pawlowski, A. Cervin, J. L. Guzman, and M. Berenguel, "Generalized Predictive Control With Actuator Deadband for Event-Based Approaches," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, 523-537, Feb. 2014.

[4] E. Moradi-Pari *et Al.*, "Design, Modeling, and Simulation of On-Demand Communication Mechanisms for Cyber-Physical Energy Systems," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2014.

[5] S. Bose, B. Natarajan, C. M. Scoglio, N. N. Schulz, D. M. Gruenbacher, and S. Das, "Shipboard Power Systems Reconfiguration—A Cyber-Physical Framework For Response Time Analysis," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, 439-449, Feb 2014.

[6] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell, "Performance Analysis of IEC 61850 Sampled Value Process Bus Networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, 1445-1454, Aug. 2013.

[7] W. Kang, K. Kapitanova, and S. H. Son, "RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, 393-405, May 2012.

[8] *IEC 61784-2 Ed. 2, "Industrial communication networks - Profiles - Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3"*, 2010.

[9] T. Sauter, "The Three Generations of Field-Level Networks—Evolution and Compatibility Issues," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, 3585-3595, Nov. 2010.

[10] P. Gaj, J. Jasperneite, and M. Felser, "Computer Communication Within Industrial Distributed Environment—a Survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, 182-189, Feb 2013.

[11] L. Zhang, H. Gao, and O. Kaynak, "Network-Induced Constraints in Networked Control Systems—A Survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, 403-416, Feb 2013.

[12] M. May, D. Ganz, and H. Doran, "HSR and PROFINET IRT bandwidth management in generic embedded systems," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Krakow, Poland, Sept. 2012.

[13] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet IO IRT Message Scheduling," in *Proc. IEEE Euromicro Conference on Real-Time Systems (ECRTS)*, Dublin, Ireland, July 2009.

[14] M. Elshuber and R. Obermaisser, "Dependable and predictable time-triggered Ethernet networks with COTS components," *Journal of Systems Architecture*, vol. 59, 2013.

[15] *IEC 61158-3-12 Ed. 2, "Industrial communication networks - Fieldbus specifications - Part 3-12: Data-link layer service definition - Type 12 elements"*, 2010.

[16] J. Jasperneite, M. Schumacher, and K. Weber, "Limits of Increasing the Performance of Industrial Ethernet Protocols," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Patras, Greece, Sept. 2007, 17-24.

[17] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Hamburg, Germany, Sept. 2008, 408-415.

[18] D. Orfanus, R. Indergaard, G. Prytz, and T. Wien, "EtherCAT-based Platform for Distributed Control in High-Performance Industrial Applications," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Cagliari. Italy, Sept. 2013.

[19] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "A Distribute-Merge Switch for EtherCAT Networks," in *Proc. IEEE International Workshop on Factory Communication Systems (WFCS)*, Nancy, France, May 2010, 121-130.

[20] G. Cena, A. Valenzano, and C. Zunino, "An arbitration-based access scheme for EtherCAT networks," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Hamburg, Germany, Sept. 2008, 416-423.

[21] G. Cena, I. C. Bertolotti, A. Valenzano, and C. Zunino, "A high-performance CAN-Like arbitration scheme for EtherCAT," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Mallorca, Spain, Sept. 2009.

[22] A. Di Stefano, A. Gangemi, L. Lo Bello, and O. Mirabella, "Slot swapping mechanisms for Process Control Networks," in *Proc. IEEE International Symposium on Industrial Electronics (ISIE)*, Guimaraes. Portugal, July 1997, 143-148.

[23] L. Lo Bello and A. Gangemi, "A slot swapping protocol for time-critical internetworking," *Journal of Systems Architecture*, vol. 51, 2005.

[24] A. Di Stefano, A. Gangemi, L. Lo Bello, and O. Mirabella, "A slot swapping based fieldbus," in *Proc. Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Aachen. Germany, Aug.-Sept. 1998, 214-219.

[25] G. Patti, L. Lo Bello, G. Alderisi, and O. Mirabella, "An EDF-based Swapping Approach to Enhance Support for Asynchronous Real-Time Traffic over EtherCAT networks," in *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Cagliari. Italy, Sept. 2013.

[26] L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, 46-61, 1973.

[27] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT Distributed Clock Performance," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, 20-29, Feb. 2012.

[28] M. Soni, *White paper: EtherCAT on Sitara TM AM335x ARM Cortex TM -A8 Microprocessors*, Texas Instruments, November 2011.

[29] S.-W. Moon, J. Rexford, and K. G. Shin, "Scalable Hardware Priority Queue Architectures for High-Speed Packet Switches," *IEEE Transaction on Computers*, vol. 49, no. 11, Nov. 2000.

[30] A. K. Mok and X. A. Feng, "Towards compositionality in real-time resource partitioning based on regularity bounds," in *Proc. IEEE Real-Time Systems Symposium (RTSS)*, London, U.K., Dec. 2001, 129-138.

[31] M. Joseph and P. K. Pandya, "Finding Response Times in a Real-Time System," *The Computer Journal*, vol. 29, no. 5, 390-395, Oct 1986.

[32] K. Gresser, "An event model for deadline verification of hard real-time systems," in *Proc. Euromicro Workshop on Real-Time Systems*, Oulu, Finland, June 1993, 118-123.

[33] K. Richter, R. Rocu, and R. Ernst, "Scheduling analysis integration for heterogeneous multiprocessor SoC," in *Proc. IEEE Real-Time Systems Symposium*, Cancun, Mexico, Dec. 2003, 236-245.

[34] S. K. Baruah, R. R. Howell, and L. E. Rosier, "Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor," *Journal of Real-Time Systems*, vol. 2, 301-324, 1990.

[35] S. K. Baruah, A. K. Mok., and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proc. IEEE Real-Time Systems Symposium (RTSS)*, Lake Buena Vista, FL, Dec. 1990.

[36] S. Lim, I.-K. Jung, and J.-H. Kim, "A performance evaluation and task scheduling of EtherCAT networked soft motion control system," in *Proc. International Symposium on Robotics (ISR)*, Seoul, Oct. 2013.

[37] S. Vitturi, L. Peretti, L. Seno, M. Zigliotto, and C. Zunino, "Real-time Ethernet networks for motion control," *Computer Standards & Interfaces*, vol. 33, 465-476, 2011.

Appendix

This Appendix provides the formal proofs of the Theorem 1 and Lemma 1 presented in Sect. III-D.

A. Theorem 1 - Derivation of the EDF schedulability conditions.

Proof: Since the priority of ApMs is assigned according to EDF, we adapt the guarantee test based on the demand bound function [34]. Differently than a classic CPU scheduling problem, the resource to be scheduled is not the CPU time but rather the aperiodic telegrams. As also illustrated by the dark gray band in Figure 4, the same aperiodic telegram is available at different times for different slaves. To remove this slave-dependent time shift, we set the common reference time at the reception side of the master (illustrated as the horizontal line at the bottom of Figure 4). The arrival of any ApM at time t at slave k can be equivalently represented by the arrival of the same message at time $t+\Delta_k$ at the master. Then, if the i-th ApM has deadline D_i from the arrival at slave π_i to the reception at the master, it can be equivalently represented by a message arriving directly at the master with deadline $D_i-\Delta_{\pi_i}$. In addition, the assumption that the latest time to transmit the ApMs is at the start of the reception of the incoming aperiodic telegram, while the reading of the aperiodic message is made only at the end of the reception of the entire frame at the master, implies that all ApM deadlines must be decremented by an amount $A=T_{et}+T_{ec}-T_{ap}$ (also represented in Figure 4 by $p\mathsf{q}+\mathsf{c}$). In conclusion, the demand bound function [34], which in this case is *the largest possible number of ApMs with arrival properly translated to the reference time at the master and deadline within an interval of length* t, is given by the following expression:

$$\sum_{i=1}^{n} \max\left(0, \left\lfloor \frac{t-(D_i-\Delta_{\pi_i}-A)}{T_i} \right\rfloor + 1\right) \tag{15}$$

Since the minimum possible number of aperiodic telegrams available in any interval of length t is represented by $s(t)$, then Condition (8) holds, because EDF can schedule any task set for which the demand bound function does not exceed the available resource for all $t \geq 0$. Hence, the Theorem is proved. ∎

B. Lemma 1- A practical schedulability condition.

Proof: We observe that $s(t)$ of (2) can be lower bounded as follows

$$\begin{aligned} s(t) &\geq \sum_{j=1}^{p}\left(\frac{t+(j-1)S}{P}-1\right) = \frac{p}{P}(t-P)+\frac{S}{P}\sum_{j=1}^{p}(j-1) \\ &= \frac{p}{P}(t-P)+\frac{S}{P}p(p-1) = \frac{p}{P}(t-(P-(p-1)S)). \end{aligned}$$

The left-hand side (LHS) of (8), which we denote for simplicity $\mathsf{dbf}(t)$ to recall the *demand bound function* of EDF scheduling [34], can be upper bounded as follows

$$\mathsf{dbf}(t) \leq \sum_{i=1}^{n} \max\left(0, \frac{t-\phi_i}{T_i}\right) = \max_{\ell=0,\ldots,n} \sum_{i=1}^{\ell} \frac{t-\phi_i}{T_i},$$

since the ordering $\phi_1 \leq \phi_2 \leq \ldots \leq \phi_n$ holds. From the lower bound of $s(t)$ and the upper bound of $\mathsf{dbt}(t)$, it follows that

$\mathsf{dbt}(t) \leq s(t)$ for all t satisfying the next condition

$$\mathsf{dbf}(t) \leq \max_{\ell=0,\dots,n} \sum_{i=1}^{\ell} \frac{t-\phi_i}{T_i} \leq \frac{p}{P}(t-(P-(p-1)S)) \leq s(t)$$

$$\forall \ell = 0,\dots,n, \quad \sum_{i=1}^{\ell} \frac{t-\phi_i}{T_i} \leq \frac{p}{P}(t-(P-(p-1)S))$$

$$\forall \ell = 0,\dots,n, \ \frac{p}{P}(P-(p-1)S) - \sum_{i=1}^{\ell} \frac{\phi_i}{T_i} \leq t(\frac{p}{P} - \sum_{i=1}^{\ell} \frac{1}{T_i})$$

$$\forall \ell = 0,\dots,n, \ t \geq \frac{\frac{p}{P}(P-(p-1)S) - \sum_{i=1}^{\ell} \frac{\phi_i}{T_i}}{\frac{p}{P} - \sum_{i=1}^{\ell} \frac{1}{T_i}}$$

$$t \geq \max_{\ell=0,\dots,n} \frac{\frac{p}{P}(P-(p-1)S) - \sum_{i=1}^{\ell} \frac{\phi_i}{T_i}}{\frac{p}{P} - \sum_{i=1}^{\ell} \frac{1}{T_i}} = L^*.$$

Note that dividing LHS and right-hand side (RHS) by $\frac{p}{P} - \sum_{i=1}^{\ell} \frac{1}{T_i}$ is possible, thanks to the hypothesis of (9).

Then $\forall t \geq L^*$ is always $\mathsf{dbf}(t) \leq s(t)$. If, instead, $t < L^*$ the condition (8) needs to be explicitly checked. However, it is sufficient to check it only at the discontinuity of the LHS, which are the absolute deadlines of all ApMs not larger than L^*, all contained in $\mathcal{D}$. ■

Lucia Lo Bello (M'02-SM'09) received the M.S. degree in electronic engineering and the Ph.D. degree in computer engineering from the University of Catania, Catania, Italy. She is an Associate Professor with tenure with the Department of Electrical, Electronic and Computer Engineering, University of Catania.

Currently she is Visiting Professor at the University of Malardalen, Sweden and also the Chair of the IES Technical Committee on Factory Automation. She authored or coauthored more than 140 technical papers in the area of industrial automation networks, real-time embedded systems and wireless sensor networks.

Enrico Bini (SM'11) received the Ph.D. degree in real-time systems from Scuola Superiore Sant'Anna, Pisa, Italy, in 2004 and two M.Sc. degrees in computer engineering and mathematics from the University of Pisa, Italy, in 2000 and 2010, respectively.

After becoming Assistant Professor at the Scuola Superiore Sant'Anna, in 2012, he spent two years at Lund University, Lund, Sweden, with a Marie-Curie Fellowship, working with Karl-Erik Årzén on Cyber-Physical Systems. While earning the Ph.D. degree, he visited Sanjoy Baruah at the University of North Carolina, Chapel Hill, NC, USA. He has published 80 papers (two best paper awards) on scheduling algorithms, real-time operating systems, and design and optimization techniques for real-time and control systems.

Gaetano Patti (M'13) received the M.S. degree (*summa cum laude*) in computer engineering from the University of Catania, Catania, Italy, in 2013.

He is currently working towards the Ph.D. degree in Systems, Energy, Computer and Telecommunications Engineering at the University of Catania.

His research interests include real-time industrial networks, Wireless Sensor Actuators Networks (WSANs), powerline communications and networks for mobile robots applications.