A Flexible Mechanism for Efficient Transmission of Aperiodic Real-Time Messages over EtherCAT networks

Lucia Lo Bello, Gaetano Patti, Giuliana Alderisi, Vito Davide Patti, Orazio Mirabella
Department of Electrical, Electronic and Computer Engineering
University of Catania
Catania, Italy
{lucia.lobello, gaetano.patti, giuliana.alderisi, orazio.mirabella}@dieei.unict.it,
vito.patti@studium.unict.it

ABSTRACT

EtherCAT is a real-time Ethernet protocol mainly designed for periodic real-time transmission in factory automation. As EtherCAT does not provide efficient mechanisms to transmit aperiodic real-time messages, recently a solution to this problem, called an EDF-based Swapping approach, was proposed. Such an approach implements a swapping policy based on the Earliest Deadline First algorithm. The approach proposed in this paper, called a Flexible EDFS, introduces two novel features. First, it reduces the overhead introduced when multiple aperiodic messages have to be transmitted. Second, it provides for variable cycle times according to the aperiodic real-time workload. The paper describes the proposed approach and presents comparative assessments, obtained through OMNeT++ simulations, with the EtherCAT standard and the original EDF-based Swapping.

I. INTRODUCTION

EtherCAT is one of the Real-Time Ethernet Communication Profiles defined in the IEC standards [1] and [2]. EtherCAT networks are characterized by a daisy-chain topology and a master/slave architecture. The master node periodically transmits standard Ethernet frames containing several *telegrams* (as shown in Fig. 1), while the slaves process the frame "on-the-fly" and read and write data in the telegrams. When a slave receives a byte, this byte is processed and transmitted to the next slave without waiting for the entire frame to arrive. The last slave in the chain has to transmit back the frame to the master and then the cycle starts again.

Thanks to the low latency and short cycle times provided, the EtherCAT protocol is suitable for real-time control applications such as, motion control and power control [3][4].

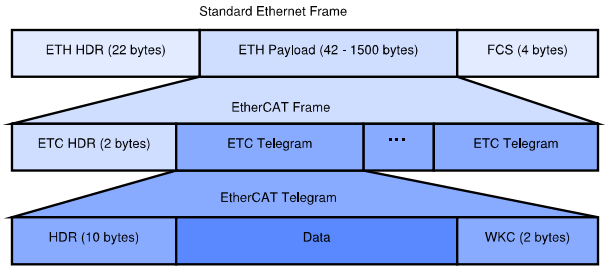

Figure 1. EtherCAT frame structure

Being a protocol specifically devised for periodic exchanges, EtherCAT does not offer specific support for aperiodic real-time traffic and schedules it in the same way as the periodic one. This means that at least one telegram has to be reserved for any slave that could require the transmission of aperiodic messages and this reservation entails an increase in the length of the EtherCAT frame. Such an increase, in turn, determines an increase in the cycle time, i.e., the time necessary to exchange the input/output data between the controller and all the networked devices once [14]. Reserving telegrams for potential transmissions that could not even occur is not only inefficient from the bandwidth point of view, but also makes the cycle time unnecessarily grow, thus reducing the support offered to applications requiring short cycle times. For this reason, a mechanism able to support aperiodic traffic transmissions without significantly increasing the cycle time is needed. In [5] an Earliest Deadline First-based Swapping (EDFS) approach was proposed that combines Earliest Deadline First (EDF) [6] scheduling with message swapping.

Swapping is possible by exploiting both the byte-to-byte frame processing provided by the EtherCAT standard and the daisy-chain topology that allow the slaves to change "on-the-fly" the telegram payload of an incoming telegram when the telegram traverses a slave.

In EDFS [5] a novel telegram type, called an aperiodic telegram, has been introduced, that is contented among the slaves according to the Earliest

Deadline First policy. Any slave with an aperiodic message to transmit enters the contention for an aperiodic telegram according to the EDF rule. Multiple aperiodic telegrams can be embedded in a single EtherCAT frame. The master will receive the aperiodic messages with the most urgent deadlines, while any message that lost the contention on its way to the master will be stored in the local queue of the slave that has swapped it, so that it will re-enter the contention whenever another aperiodic telegram will traverse that slave.

In [5] the EDFS performance was evaluated in terms of cycle time. Results showed that EDFS is able to support real-time aperiodic transmissions while reducing the cycle time comparing with the EtherCAT standard.

This paper proposes a new swapping-based approach, called Flexible EDFS (F_EDFS), which introduces two new properties comparing with EDFS. First, it allows multiple aperiodic messages to be embedded in a single aperiodic telegram, thus significantly reducing the overhead introduced by the header and trailer of each aperiodic telegram in EDFS. Moreover in F_EDFS the length of the aperiodic telegram is dynamically controlled by the master so that the number of aperiodic messages sent in a cycle varies according to the number of slaves that have aperiodic real-time messages to transmit. This property is valuable, as a lower aperiodic real-time workload will entail a reduction of the cycle time. A shorter cycle time entails lower response times, thus increasing the number of aperiodic messages delivered within their deadlines. The maximum number of aperiodic messages which can be transmitted in a cycle is defined in the configuration phase, so an upper bound on the maximum cycle time is guaranteed.

The paper is organized as follows. Section 2 outlines related work. Section 3 introduces the F_EDFS approach, while Section 4 presents a comparative performance assessment between the F_EDFS, the EDFS, and the EtherCAT standard through OMNeT++ simulations. Finally, Section 5 concludes the paper and outlines directions for future work.

II. RELATED WORK

Many works in the literature addressed the performance of the EtherCAT standard in several industrial application domains [7] [8] [9].

With reference to the specific problem addressed in this paper, i.e., how to efficiently support the transmission of aperiodic real-time messages, in [10] an approach is proposed, that will be henceforward called CAN-Like, in which a telegram is contented among slaves for the transmission of aperiodic data. Arbitration is performed, similarly to CAN networks, through a byte-to-byte comparison of the message identifiers. In the CAN-Like approach the transmission of one aperiodic message requires two cycles: The first one is for the transmission of the aperiodic message and the second for the "acknowledgement" of the message received by the master.

In [11] an improvement of the CAN-Like was proposed, which enables to embed multiple aperiodic messages within a single telegram. The approach proposed in [11] still suffers from some limitation, as two cycles are required to transmit messages. In fact, in [11] the overwritten message is lost, so the slaves that have transmitted an aperiodic message do not know if their message has been overwritten or delivered to the master. Consequently, the slaves need a cumulative "ack" message from the master to be informed about either the reception of their message or the need for retransmitting it.

Unlike the previously described approaches, the EDF-based Swapping (EDFS) proposed in [5] and inspired by the Slot Swapping Protocol described in [12], allows slaves to transmit aperiodic messages without the need for an "acknowledgement" from the master. The reason is that in EDFS the slave that has swapped an incoming message stores such a message in its local queue. As a result, there is no need for the slave that had sent the swapped message to re-send it, as the message has not been overwritten but simply swapped, so it will be up to the slave that has swapped it to try and send it whenever possible. Moreover, the approach in [5] also provides the possibility to embed multiple aperiodic telegrams in one EtherCAT frame.

In the EDFS a novel telegram is defined, that is called an aperiodic telegram. Such a telegram is contended among slaves according to the EDF rule [6], i.e., the message with the closest absolute deadline is the message with the highest priority and so wins the contention. The aperiodic message is encapsulated in a packet called APDU. In Fig. 2 the structure of the aperiodic telegram and the APDU is shown.

The *APDU_DL* field encodes the absolute deadline of the message as the number of microseconds elapsed since January 1, 2000 (i.e., the system time [1]).

Each slave maintains a local queue of APDUs ordered according to their absolute deadlines. When a slave with an APDU to transmit (M_{loc}) is traversed by an aperiodic telegram containing an APDU (M_{in}), the slave compares byte-to-byte the deadline of M_{in} with the one of M_{loc}. If M_{loc} has a closer deadline than M_{in}, the slave transmits M_{loc}. At the end of the transmission,

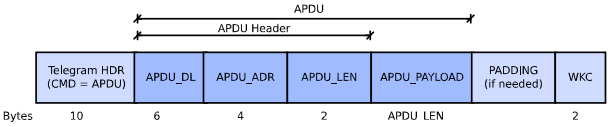

Figure 2. Aperiodic telegram structure of the EDF-Based Swapping

M_{in} is inserted by the slave in its local queue (swapping), so that it can be transmitted in another aperiodic telegram.

The EDFS allows transmission of aperiodic real-time messages while maintaining short cycle times. The number of aperiodic telegrams to be embedded in an EtherCAT frame can be selected according to the workload of aperiodic messages as shown in [5].

In the original EDFS the number of aperiodic telegrams is set at configuration time, so the cycle time is constant independently of the actual aperiodic workload. In [5] the cycle times achieved by EDFS are found significantly shorter than the ones obtained by the EtherCAT standard.

As it will be explained in the next Section, the F_EDFS approach introduced in this paper introduces a further property, i.e., the cycle time is not fixed for all the frames, but varies according to the actual real-time aperiodic traffic demand, so that the lower the aperiodic workload, the shorter the F_EDFS cycle time. This property allows for a higher number of aperiodic messages delivered within their deadlines (especially in the case of low aperiodic workload), as shorter cycle times entail lower response times for the aperiodic messages.

Moreover, in EDFS each aperiodic telegram introduces a fixed overhead due to the header and trailer of each telegram. As it will be shown in the following of this paper, the reduction of such an overhead in the F_EDFS approach here proposed also contributes to improve the network performance in terms of cycle time, thus reducing the response times of the aperiodic real-time messages even further.

III. THE FLEXIBLE EDF-BASED SWAPPING

The Flexible EDFS here proposed provides two novel features. Firstly, the possibility for the slaves to transmit multiple APDUs in a single aperiodic telegram is introduced. Secondly, a mechanism for dynamically varying the size of the aperiodic telegram (hence the number of APDUs that can be embedded in it) is presented. Such a mechanism takes into account the number of slaves which have an APDU to transmit. As the reduction of the aperiodic telegram length entails shorter cycle times, under a low aperiodic workload reducing the number of APDUs transmitted within a cycle improves the protocol performance.

The basic idea of F_EDFS is to partition the aperiodic telegram in multiple segments of the same length, so that an APDU for each segment can be transmitted. The length of a segment (L_{max}) is off-line chosen taking as a reference the largest APDU which can be transmitted, thus ensuring enough space for any APDU, while the length of the aperiodic telegram (hence the number of segments) varies every cycle according to the number of slaves that have an APDU to transmit. The information about the number of slaves that have an APDU to transmit is encoded in the working counter field (WKC) of the aperiodic telegram, whose structure is shown in Fig. 3.

In order to embed multiple APDUs in one aperiodic telegram, the length of the aperiodic telegram must be chosen as a multiple of L_{max} so as to contain $N \cdot L_{max}$ APDUs. To mark the position at which a segment starts, in every cycle the master transmits an aperiodic telegram containing special APDUs representing the points in the telegram where a slave can transmit its APDU. Such special APDUs have the deadline field (APDU_DL) equal to 0xFFFFFFFFFFFF, so any slave can easily recognize them. A slave that has an APDU to transmit overwrites the special APDU transmitted by the master with its local APDU. The next slave that has an APDU to transmit will read the APDU_DL field and then will start the arbitration. In this way, slaves do not need additional fields to know where an APDU begins within the aperiodic telegram.

Moreover, when the slaves with an APDU to transmit are traversed by an aperiodic telegram, both in the case they win or lose the contention, they increment the working counter field by 1, so that the master, when receiving the EtherCAT frame, also gets a feedback about the number of slaves that need to transmit an APDU. In this way the master is able to adapt the size of the aperiodic telegram to the needs of aperiodic real-time traffic.

The aim of such a mechanism is to reduce the average cycle times, while meeting the maximum cycle time constraint imposed by the application and set at configuration time. The idea is to reduce the number of APDUs which can be transmitted in a cycle when the aperiodic workload is low and to increment the aperiodic telegram length when the workload grows, but without exceeding the maximum cycle time allowed by the application. The adaptive algorithm here proposed works as follows:

- In the first cycle, the master transmits an aperiodic telegram of length N L_{max}, with N chosen so as to have the maximum allowed cycle time.
- In the other cycles the master calculates the difference (z) between the working counter and the number N of APDUs which can be transmitted in the aperiodic telegram, ($z = wkc - N$).
- If z is equal to 0, N is not changed.

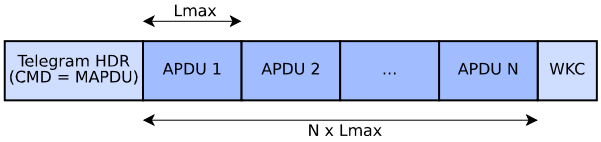

Figure 3. Structure of the aperiodic telegram containing multiple APDUs

- If z is lower than 0 (that is, no slaves have an APDU to transmit), z has a negative value and N is decreased by 1 (with a minimum value of $N=1$).
- If z is higher than 0, N is increased by 1 (up to a maximum value of N chosen so as to not exceed the maximum cycle time set at configuration time).

As it will be shown in Sect. 4.2 F_EDFS largely outperforms the EtherCAT standard in terms of cycle times. In addition, thanks to its adaptive mechanism, F_EDFS also obtains better cycle time values than the EDFS, for two reasons. First, F_EDFS avoids the 12-byte overhead introduced by the telegram header and working counter for each APDU transmission. In fact, in the case of multiple APDU transmissions, in the original EDFS multiple aperiodic telegrams are required, and so such an overhead is introduced per each APDU transmission. On the contrary, in F_EDFS multiple APDUs are embedded in one aperiodic telegram and only one aperiodic telegram is transmitted in each cycle, so the overhead is introduced only once per each cycle. Second, as in F_EDFS the number of APDUs which can be embedded in an aperiodic message is dynamically changed according to the actual aperiodic workload, the mean cycle time is also improved.

The novel mechanisms introduced in F_EDFS can be implemented with minor changes in the EtherCAT protocol. As in EDFS, a novel telegram specifically devised for aperiodic messages has to be introduced.

A local queue has to be implemented in the data link layer of the slaves to contain both the local and the swapped APDUs, ordered according to their absolute deadlines. The arbitration procedure based on the deadline comparison is a simple operation that can be realized via software or hardware implementation.

IV. SIMULATION SCENARIO AND RESULTS

In this section the Flexible EDF-based Swapping (F_EDFS) is compared with the original EDF-based Swapping (EDFS) and the EtherCAT standard. A suitable simulation model was implemented with the OMNeT++ framework [13]. The simulation model implements two kinds of nodes:

- *EtherCAT Master:* this module emulates the master. It periodically transmits an Ethernet frame supporting the F_EDFS and collects statistics.
- *EtherCAT Slave:* this module emulates the slave. It is composed of three simple modules, i.e., the traffic generator, the DLL, and the PHY module. In particular, the DLL module is the core of the simulation model, as it implements the contention policies of the F_EDFS (i.e., the local queue of APDUs, the contention algorithms, etc…).

In the simulation model the cable propagation delay is not considered, as this parameter affects both the EDFS and the F_EDF in the same way and depends on the cable length between the nodes.

The simulation model was assessed by comparing the cycle times of the simulated network with the analytical ones calculated using formula (1) found in [14] and [7],

$$T_c = T_{eth} + T_{etc} + L\,T_{to} + \sum_{i=1}^{L} T_{ct}^{(i)} + M\,T_{sv} + T_{if} \quad (1)$$

where

- T_{eth} is the time necessary for the transmission of the Ethernet header and Frame Check Sequence (FCS).
- T_{etc} is the time for the transmission of the EtherCAT header.
- L is the number of telegrams.
- T_{to} is the transmission time of the telegram header and working counter.
- $T_{ct}^{(i)}$ is the time necessary for the transmission of the payload of the *i-th* telegram.
- M is the number of slaves.
- T_{sv} is the slave processing time for the Ethernet frame.
- T_{if} is the interframe gap.

In F_EDFS the cycle time may vary from one cycle to the next with the number of APDUs contained in the j-th cycle aperiodic telegram.

I.1. Simulated scenario

The aim of the simulations is to compare the performance of the three approaches here considered in terms of cycle time, APDUs response time and deadline miss ratio.

In the simulated scenario an EtherCAT network with 10 slaves was deployed, as shown in Fig. 4.

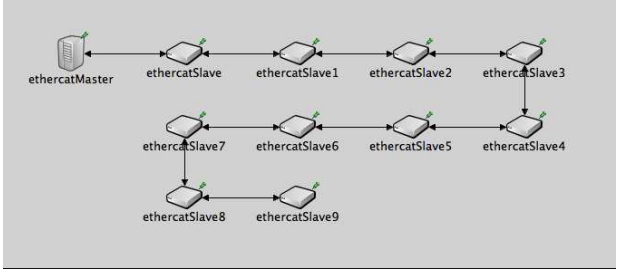

Figure 4. Simulated Network

The master cyclically transmits an Ethernet frame containing both periodic and aperiodic telegrams.

In all the simulations the periodic traffic is constant and low (i.e., 2 telegrams with 16-byte payload), as the purpose of this assessment is to evaluate the effect of increasing the aperiodic real-time workload. Aperiodic messages are generated by slaves according to an exponentially distributed interval with mean λ. Table 1

summarizes the notation used for the simulation parameters, while Table 2 shows the values chosen for the simulation parameters.

Parameter	Symbol
Number of slaves	M
Periodic payload size	S_{pt}
Periodic telegrams number	N_{pt}
Aperiodic telegram size	S_{at}
Aperiodic telegram number	N_{at}
Slave latency	T_{sv}
APDU relative deadline	D_i
Mean generation interval for APDUs	Λ
Network mean aperiodic workload	W

Table 1. Notation used for the simulation parameters

Parameter	Value/range	
	EDFS	F_EDFS
M	10	10
N_{pt}	2	2
S_{pt}	16 bytes	16 bytes
N_{at}	4	1
S_{at}	28 bytes	from 28 to 112 bytes
T_{sv}	700 ns	700 ns
λ	75µs, 82µs, 164µs	75µs, 82µs, 164µs
w	133333, 121951, 60975 APDU/s	

Table 2. Simulation parameters

In the EDFS simulation the number of aperiodic telegrams (i.e., 4) was chosen heuristically, according to the results found in [5], so as to deliver all the aperiodic workload generated while maintaining a low cycle time comparing with the EtherCAT standard.

Each APDU was generated with a random relative deadline (D_i) chosen, according to a uniform distribution, between one of the following values: 400µs, 800µs, and 1200µs. The absolute deadline of each APDU was determined according to its generation time and relative deadline. In our model, the messages that miss their deadlines are dropped, as here we assume that expired messages bring no value from the application perspective.

Three aperiodic workload scenarios were simulated by varying the λ value. The highest workload value (i.e., λ=75µs) has been chosen taking into account the workload achievable by EDFS without saturating the local queue of the slaves. Based on this value, the λ values for the medium and low workload were chosen accordingly. In order to obtain statistically significant results, the network simulation time was tuned to collect statistics from at least 50000 APDUs. Three different simulations, each one characterized by different workload patterns, were run. The simulation was repeated 5 times varying the seed of the pseudorandom number generators.

The assessed performance indicators are:

- The mean and maximum cycle time (T_c) of the F_EDFS, the EDFS, and the EtherCAT standard, respectively.
- The Deadline Miss Ratio (DMR), defined as the number of APDUs that missed their deadlines over the total number of APDUs generated.
- The response time (R) of APDUs, measured at the application layer.

In the following subsections the results obtained via simulation, for the EDFS, the F_EDFS and the EtherCAT standard are presented.

I.2. Cycle time comparison

In Table 3 the cycle times obtained by the EDFS, the F_EDFS and the EtherCAT standard are compared. The workload patterns are identified by the λ value in the first column of the Table.

The cycle time of the EtherCAT standard shown in the second column of Table 3 was obtained by reserving in the EtherCAT frame a telegram with a 28-byte payload telegram for each slave.

In the EDFS scenario we set all the EtherCAT frames to the same size, so that each EtherCAT frame contains four aperiodic telegrams. For this reason the cycle time is the same for every EtherCAT frame received, i.e., 27.4µs, irrespectively of the workload pattern, as shown in the third column of Table 3.

Conversely, the cycle time obtained with the F_EDFS is not constant, as the number of APDUs in the same EtherCAT frame is not fixed and so the length of the entire EtherCAT frame changes. The maximum cycle time obtained by F_EDFS is always lower than that obtained by both EDFS and the EtherCAT standard. F_EDFS obtains a 28% reduction of the cycle time comparing with EDFS, under the lowest workload (i.e., λ = 164µs). This reduction decreases under a higher workload, but it always remains in the order of 15%. Compared with the EtherCAT standard the cycle time improvement is in the range of [41-48]%.

λ (µs)	Cycle Time (µs)			
	EtherCAT std	EDFS	F_EDFS T_c	
			Mean	Max
75	46.68	27.40	23.2	24.6
82	46.68	27.40	22.7	24.6
164	46.68	27.40	19.6	24.6

Table 3. Cycle Time Comparison

Under high workload, the slight difference in terms of mean cycle time between the EDFS and the F_EDFS is thanks to the overhead reduction. In fact, with F_EDFS a single telegram containing up to four

APDUs is sent instead of four separate aperiodic telegrams. This means that the overhead due to the telegram header and trailer is introduced only once.

I.3. Deadline Miss Ratio comparison

In Table 4 the deadline miss ratio (DMR) values, expressed as a percentage of the overall number of APDUs generated, are shown for both the F_EDFS and EDFS approaches. The DMR experienced by aperiodic messages with the EtherCAT standard is always zero, as each slave is reserved one periodic telegram for the transmission of aperiodic messages. For this reason, here we compare EDFS and F_EDFS only. The DMR values obtained with F_EDFS, shown in the third column, are one order of magnitude lower than the results of EDFS. The worst DMR value for EDFS is obtained under high workload, (i.e., with λ equal to 75μs): In this case, about 1% of the APDUs are dropped as they missed their deadlines. The highest DMR value obtained by EDFS refers to the highest workload and it is equal to 0.1%. Under low workload (i.e., λ=164μs) the DMR obtained by both F_EDFS and EDFS is null.

λ (μs)	DMR (%)	
	EDFS	**F_EDFS**
75	≈ 1	≈ 0.1
82	≈ 0.1	≈ 0.02
164	0	0

Table 4. Deadline Miss Ratio comparison EDFS vs. F_EDFS

I.4. Response time comparison

Table 5 shows the APDUs mean response times obtained by EDFS (second column) and F_EDFS (third column), respectively. Under high workload the results obtained by F_EDFS are significantly lower than those obtained by EDFS, while in the other cases the results are comparable. Figure 5 shows the histograms representing the percentage of packets that are received by the master with a given response time.

λ (μs)	Mean Response time (μs)	
	EDFS	**F_EDFS**
75	42.5	35
82	33.5	31.3
164	21.2	23.3

Table 5. Response time comparison

For the sake of brevity, for each workload pattern the results relevant to the packets with the shortest relative deadline, i.e. 400μs, are shown. The results obtained with the other relative deadline values show similar trends. In Figure 5 the dark-colored bar represents the results obtained by F_EDFS, while the light-colored bar shows the ones obtained by EDFS. The results are shown as a percentage of all the packets received by the master, as this number slightly changes from one scenario to another. No considerable

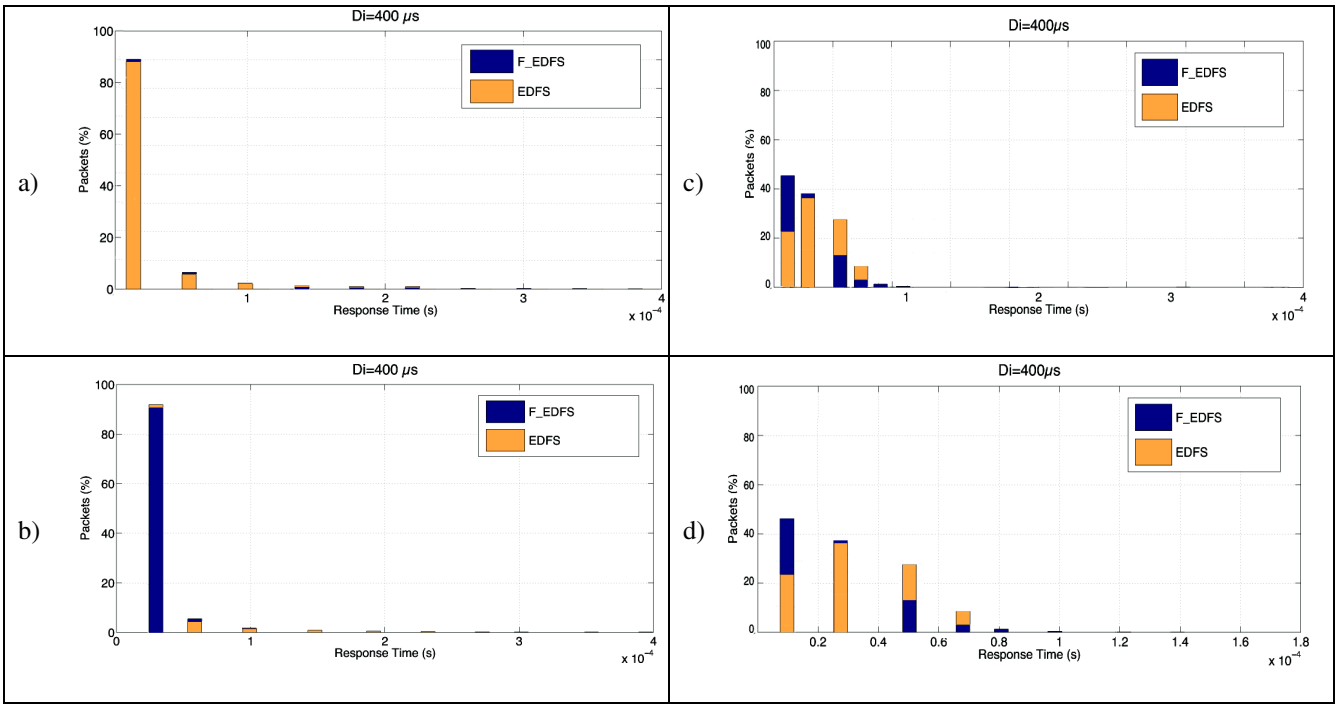

Figure 5. Histograms showing the percentage of received packets vs. response time under (a) high workload; (b) medium workload; (c) low workload; (d) detail of (c).

differences between EDFS and F_EDFS are found as far as the high and medium workload patterns are concerned, as shown in Figures 5a and 5b.

Regarding the low workload scenario shown in Fig. 5c, the F_EDFS approach reduces the response time of the aperiodic messages. As shown in Figure 5d, that provides the same results as Fig. 5c but in more detail, about the 90% of the packets with relative deadlines equal to 400μs experience a response time between 10 and 30μs. This is thanks to the variable number of segments in a single aperiodic telegram. Tables 6a-6c detail the results shown in the histograms, also providing the relevant confidence intervals, as the values shown in Figures 5 are the mean of the values obtained in the five simulations. The confidence interval is the interval in which there is the 95% probability to find the mean values given in the first and the third column of Tables 6a-6c.

a)

EDFS Received Packets (%)	Confidence Interval (%)	F_EDFS Received Packets (%)	Confidence Interval (%)
24.2	±0.08	48.7	±0.16
36.5	±0.08	39.9	±0.08
29.5	±0.07	8.58	±0.06
8.91	±0.12	1.62	±0.009
0.52	±0.008	0.65	±0.005
0.13	±0.0009	0.24	±0.002
0.07	±0.001	0.1	±0.001
0.02	±0.0002	0.01	±0.0001
0.008	±8.77e-05	0.02	±0.0003
0.01	±0.0001	0.01	±6.42e-05

b)

EDFS Received Packets (%)	Confidence Interval (%)	F_EDF Received Packets (%)	Confidence Interval (%)
91.8	±0.001	91.06	±0.001
4.3	±0.001	5.45	±0.001
1.48	±0.0007	1.77	±0.001
0.82	±0.0005	0.78	±0.001
0.52	±0.0003	0.39	±0.0001
0.34	±0.0002	0.21	±0.00015
0.22	±0.0004	0.13	±0.0001
0.18	±0.0002	0.08	±0.0001
0.13	±0.0001	0.04	±0.0001
0.09	±0.0001	0.04	±6.06 e-05

c)

EDFS Received Packets (%)	Confidence Interval (%)	F_EDFS Received Packets (%)	Confidence Interval (%)
88.8	±0.0006	89.1	±0.0009
5.1	±0.0007	5.79	±0.001
2	±0.001	2.06	±0.0004
1.1	±0.0002	1	±0.00038
0.84	±0.0005	0.68	±0.00035
0.59	±0.0006	0.43	±0.00035
0.43	±0.0003	0.27	±0.00035
0.38	±0.0004	0.2	±0.00035
0.3	±0.0003	0.15	±0.00035
0.3	±0.0001	0.12	±0.00035

Table 6. Percentage of packets received vs. response time under: (a) low, (b) medium, and (c) high workload.

V. CONCLUSIONS AND FUTURE WORK

This work addressed the problem of providing an efficient support for aperiodic real-time messages over EtherCAT networks and proposed the Flexible EDF-based Swapping approach. The ability of swapping messages combined with EDF scheduling is a very beneficial feature that allows for a significant reduction of the cycle time over a broad range of aperiodic workloads.

EDF-based swapping can be introduced in the EtherCAT protocol with minimal changes, i.e., the introduction of: a novel telegram specifically devised for aperiodic messages, the arbitration procedure based on the comparison between the deadline fields of the competing APDUs, and a local queue at each slave, ordered according to EDF, to store the swapped messages.

Compared with EDFS, that was the first approach in the literature proposing EDF-based swapping in EtherCAT networks, the F_EDFS introduces two novel features, i.e., the possibility to embed multiple aperiodic messages within a single aperiodic telegram and the ability to adapt the number of aperiodic messages contained in an aperiodic telegram to the real-time aperiodic workload. The adaptation algorithm exploits the knowledge of the number of slaves that have an APDU to transmit (as each slave increments the working counter field by one when is traversed by a frame) and is able to provide an upper bound on the maximum cycle time. In fact, the maximum number of aperiodic messages which can be transmitted is defined in the configuration phase depending on the application requirements.

The performance of F_EDFS was assessed through simulations in three different scenarios, i.e., under high, medium, and low workload. Simulation results showed for the F_EDFS a reduction of the mean cycle times up to 20% compared with the EDFS and up to 48% compared with the EtherCAT standard. As far as the deadline miss ratio is concerned, a reduction of one order of magnitude was found with F_EDFS comparing with EDFS.

Summarizing, under low workload F_EDFS obtains significantly better results than the other two

approaches. Under high workload, the response time results of F_EDFS and EDFS are comparable, while the F_EDFS cycle time values (both mean and maximum values) are lower than those obtained by both the EDFS and the EtherCAT standard. Moreover, the deadline miss ratio is also significantly improved by F_EDFS comparing with EDFS.

Future work will deal with a framework for the F_EDFS schedulability analysis enabling the network designer to calculate the response times of aperiodic messages under given workload conditions.

REFERENCES

[1] IEC 61158-3/4-12 Ed. 2, "*Industrial communication networks - Fieldbus specifications*", 2010.

[2] 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.

[3] Xuepei Wu, Lihua Xie, F. Lim, "EtherCAT-enabled next generation Baggage Handling Systems", in *Proc. IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, Cagliari, Italy, Sept. 2013.

[4] W. Zheng, H. Ma, X. He, "Modeling, analysis, and implementation of real time network controlled parallel multi-inverter systems" in *Proc. International Power Electronics and Motion Control Conference (IPEMC)*, Harbin, China, June 2012, pp. 1125–1130.

[5] G. Patti, L. Lo Bello, G. Alderisi, 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.

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

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

[8] 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.

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

[10] G. Cena, A. Valenzano, 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, pp. 416-423.

[11] G. Cena, I. C. Bertolotti, A. Valenzano, 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.

[12] L. Lo Bello, A. Gangemi, "A slot swapping protocol for time-critical internetworking", Journal of Systems Architecture, vol. 51, pp. 526–541, March 2005.

[13] OMNeT++ Simulation Framework, v. 4.3.1 [online]. Available http://www.omnetpp.org.

[14] J. Jasperneite, M. Schumacher, 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, pp. 17-24.