Schedulability Analysis of Real-Time Traffic in WorldFIP Networks: An Integrated Approach

Luís Almeida, *Member, IEEE*, Eduardo Tovar, *Member, IEEE*, José Alberto G. Fonseca, *Member, IEEE*, and Francisco Vasques, *Member, IEEE*

***Abstract*—The WorldFIP protocol is one of the profiles that constitute the European fieldbus standard EN-50170. It is particularly well suited to be used in distributed computer-controlled systems where a set of process variables must be shared among network devices. To cope with the real-time requirements of such systems, the protocol provides communication services based on the exchange of periodic and aperiodic identified variables. The periodic exchanges have the highest priority and are executed at run time according to a cyclic schedule. Therefore, the respective schedulability can be determined at pre-run-time when building the schedule table. Concerning the aperiodic exchanges, the situation is different since their priority is lower and they are handled according to a first-come–first-served policy. In this paper, a response-time-based schedulability analysis for the real-time traffic is presented. Such analysis considers both types of traffic in an integrated way, according to their priorities. Furthermore, a fixed-priorities-based policy is also used to schedule the periodic traffic. The proposed analysis represents an improvement relative to previous work and it can be evaluated online as part of a traffic online admission control. This feature is of particular importance when a planning scheduler is used, instead of the typical offline static scheduler, to allow online changes to the set of periodic process variables.**

***Index Terms*—Distributed computer control systems, real-time communication, response time, scheduling algorithms.**

I. Introduction

THE use of fieldbuses in distributed industrial control applications has become widespread in recent years. This is due, among other factors, to the advantages arising either from a more manageable and less expensive cabling system as well as from an increased system robustness associated to the distribution of system intelligence. In distributed control applications, the fieldbus plays a fundamental role interconnecting the field devices (e.g., sensors, actuators, and controllers) to form distributed control loops [20]. Real-time operation is then necessary to ensure timely transmission of information.

Therefore, a traffic schedulability analysis is required in order to guarantee a timely behavior of the fieldbus network. Although there is a common background for most of such analysis [5], [11], [14], its use in a particular bus network protocol always requires specific adaptations (e.g., [21]).

Manuscript received November 3, 2000; revised April 23, 2002. Abstract published on the Internet July 15, 2002.

L. Almeida and J. A. G. Fonseca are with the DET—IEETA, Universidade de Aveiro, P-3810-193 Aveiro, Portugal (e-mail: lda@det.ua.pt; jaf@det.ua.pt).

E. Tovar is with the DEI—ISEP, Instituto Politécnico do Porto, Porto, Portugal (e-mail: emt@dei.isep.ipp.pt).

F. Vasques is with the DEMEGI—FEUP, Universidade do Porto, Porto, Portugal (e-mail: vasques@fe.up.pt).

Publisher Item Identifier 10.1109/TIE.2002.803242.

A widely used fieldbus standard is WorldFIP [6], which is one among the profiles that constitute the European fieldbus standard EN-50170. Basically, WorldFIP delivers two types of communication services: exchanges of identified variables and exchanges of messages. Real-time communication services are based upon the former type. The latter is used to support manufacturing message services (MMS) [10] and is out of the scope of this paper. Concerning the exchanges of identified variables, WorldFIP separates these in two major categories: periodic and aperiodic exchanges. The periodic exchanges are triggered by the communication system itself and have the highest priority. On the other hand, aperiodic exchanges occur in response to dynamic requests issued by application tasks through application layer services.

There are two different levels of priority for aperiodic exchanges: urgent and normal. Although urgent exchanges are processed before normal ones, both levels are processed only when there are no periodic exchanges waiting to be performed. Moreover, aperiodic requests are brought to the knowledge of the communication system by means of the periodic exchanges issued by each node and are handled in a first-come–first-served (FCFS) order. These characteristics impose relatively large response times to the aperiodic exchanges when compared to the periodic ones.

Building upon recent work carried out by the authors [3], [4], [23], [24], this paper presents a response-time analysis for both periodic and aperiodic exchanges of identified variables. Based on such analysis, an integrated schedulability condition is derived that can be used for both types of exchanges. Under certain assumptions, such condition is necessary and sufficient for the periodic traffic although just sufficient for the aperiodic one. The scheduling policy used for periodic traffic is based on fixed priorities.

The proposed analysis is particularly relevant when the original WorldFIP's static table-based scheduling of periodic traffic is replaced by a dynamic table-based scheduler, such as the planning scheduler [2]. In such case, it is possible to introduce online modifications to the set of periodic exchanges. This analysis is well suited to support an online schedulability analyzer tool, which can be used to validate modifications on any set of exchanges, either periodic or aperiodic.

This paper is organized as follows. Section II presents a brief overview of the WorldFIP protocol together with a discussion of existing related work. Section III presents an analysis of the basic transactions performed by WorldFIP. Section IV establishes the computational model and analyzes the worst case scenarios for both periodic and aperiodic transactions. In Section V, the timeline analysis is used to derive upper bounds to the response times to communication requests. It is first applied to the periodic exchanges and then enlarged to integrate aperiodic exchanges. Section VI applies the timeline analysis to a benchmark scenario, and Section VII draws some concluding remarks.

II. Overview and Related Work

The WorldFIP protocol was developed in the early 1980s [8] (formerly known as FIP—Factory Instrumentation Protocol) and later established as a French standard [1]. In the meantime, much research work has been conducted concerning various aspects of that protocol, e.g., modeling and formal verification, data coherence issues either in time as well as in space, network interconnection through bridges, etc. However, due to their relevance in the scope of the paper, only the construction of the bus arbitrator table and the analysis of the response time to communication requests will be considered here. These two aspects are closely related since the particular criterion used to build the bus arbitrator table has a deep influence on the system temporal behavior. Nevertheless, before going through such aspects, a brief overview of the underlying communication mechanisms will be presented.

A. Exchanging Identified Variables

The MAC protocol of WorldFIP relies on centrally controlled access and the underlying communication model is the producers–distributor–consumers (PDC) model [19]. Hence, a master node, i.e., the bus arbitrator (BA), starts all the transactions, sequentially, by transmitting a frame with the identification of a given variable (distributor role). The node that produces such variable (producer) responds transmitting a frame with the corresponding data. Simultaneously, all nodes requiring the value of that variable (consumers) copy the data contents of the frame to local buffers (accomplishing the variable exchange). This is how a periodic exchange is carried out in WorldFIP. These exchanges are also used by the producers to signal the BA that there is a pending aperiodic request.

A static schedule table, known as a BA table or, simply, BAT, supports the orderly release of the periodic transactions. The BAT is organized as a sequence of elementary cycles (ECs). Each EC refers to a bus time window and contains the identification of all the periodic exchanges that must be processed during that window. Within each EC there cannot be more than one exchange of each identified variable. Thus, the periods of all the periodic exchanges are expressed as an integer number of ECs.

The traffic cyclic pattern, which results from the periodic nature of the exchanges, has a period equal to the least common multiple (LCM) of the variables periods and is called the macrocycle (MC). The BAT contains the sequence of ECs that constitute the MC (Fig. 1).

The WorldFIP fieldbus has several modes of operation in what concerns the synchronization of the ECs and the MC. The most common mode is the synchronous one, in which the ECs always have a constant duration established at configuration time (timer T2), being thus possible to assure periodic

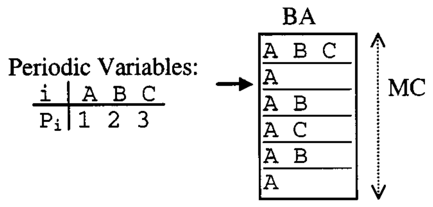

Fig. 1. BAT and respective ECs for the set of periodic variables shown.

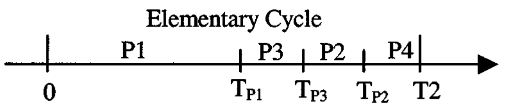

Fig. 2. EC, composed of four consecutive phases.

exchanges with low jitter. If the exchanges to be performed within one EC take less time than the EC duration, the BA transmits padding frames up to the end of that EC. When this synchronous mode is used, the EC duration is typically set to the highest common factor (HCF) of the variables' scanning periods.

Each EC can contain up to four consecutive windows, the periodic window (P1), the messages window (P2), the urgent and normal aperiodic window (P3), and a synchronization window (P4). The duration of phase P1 $(T_{\rm P1})$ can be indirectly constrained when building the BAT, by limiting the number of transactions scheduled for any EC. Phases P2 and P3 can be processed in either order setup, and their maximum durations, referred to the beginning of the EC ($T_{\rm P2}$ and $T_{\rm P3}$) can be directly specified at configuration time. Finally, phase P4 includes the possible transmission of padding frames, as required for synchronous mode operation. This is depicted in Fig. 2.

Without loss of generality, the remaining discussion will not consider phase P2 (messages), only P1, P3, and P4. Furthermore, it will be considered that the periodic window (P1) may take the full EC duration $(T_{\rm P1} = {\rm T2})$. Whenever it takes less, the aperiodic window can use the full remaining time $(T_{\rm P3} = {\rm T2})$.

B. Previous Relevant Work on Building the BAT

In typical WorldFIP applications, the BAT is built offline and thus, computationally intensive scheduling algorithms can be used. It is possible, this way, to use traditional operations research techniques, such as linear programming, to find a schedule that optimizes a given parameter, e.g., minimizes the maximum response time, minimizes the maximum release jitter, maximizes the load balance in all ECs. For example, Dworzecki [7] proposes one such scheduler that is claimed to be more efficient than normal rate-monotonic (RMS) or earliest deadline first (EDF) for building WorldFIP BATs.

On the other hand, Tovar and Vasques [22] show examples of using priority-based techniques such as RMS and EDF [14]. Raja and Noubir [18] also consider these techniques in a FIP-like system but under the assumption that all bus transactions take the same time, i.e., the duration of one slot. Still under the one slot assumption, Kumaran and Decotignie [13] suggest the uniform distribution (UD) algorithm according to which variables with the same period are scheduled with a constant relative phase increment of one EC. This dephasing allows obtaining a smooth distribution for the periodic communication load among the ECs that form the MC.

However, most of the work mentioned before dealt with the construction of the BAT, not considering its impact on the aperiodic traffic. In particular, it is not always easy to determine the point in time, within the BAT, that represents the start of a window with highest cumulative periodic load, so that the interference caused to the aperiodic traffic is maximized. This is normally referred to as a *critical instant* for the aperiodic traffic. Its determination in BATs built with most of the above methods generally requires an extensive search along the whole table. On the other hand, when the BAT is built based on fixed priorities scheduling, the critical instant for the aperiodic traffic is well defined and coincides with the point in time where all periodic variables become ready to be transmitted simultaneously. Therefore, in order to facilitate the analysis of the aperiodic traffic, it is considered in this paper that the periodic traffic is scheduled according to a set of fixed-priorities.

Kim *et al.* [12] address two other aspects that deserve particular attention when building the BAT: memory requirements and release jitter. The first aspect is related to the potential memory size problem that might result from scheduling variables with relative prime scan periods. In this case, the LCM of such periods is potentially very large requiring large amounts of physical memory to hold the BAT. The proposed solution is to reduce the larger scan periods avoiding relative prime values. This, nevertheless, increases unnecessarily the network utilization. Concerning the second aspect, the release jitter is due to variations in the interference caused by higher priority variables, which generally differs from occurrence to occurrence. The proposed solution is to reduce the scan periods up to the point where they all become harmonic in powers of 2, i.e., being T_1 the shortest period, then $T_i = 2^{k_i} * T_1, \forall_{i=2,\ldots,N}$, for some k_i integer. In this case, it is possible to generate a jitter-free schedule. This solution, also proposed by Hong [9] in a more general context, may lead to a substantial increase in the network utilization, well beyond the application requirements. The perspective used by Hong is to deduce a set of harmonic scan periods (in powers of 2) and relative phases that fulfill the application requirements and maximizes the network utilization. In other words, the idea is to use all the network bandwidth available to facilitate meeting the application requirements.

The table size problem has also been addressed by the authors in previous work [2], where the use of a dynamic table to schedule periodic transactions has been proposed. This technique, known as the planning scheduler, consists on building a schedule, online, for a fixed period of time into the future called a plan, and constantly rebuilding the schedule, plan after plan. Notice that the plan duration is fixed and not related with the variables' scanning periods. The advantage of using the planning scheduler goes beyond the confinement of the BATs memory requirements. For example, since the scheduler is working online, it is possible to perform modifications to the periodic communication requirements while the system is running. This results on an increased operational flexibility of the protocol. It has also been shown that the planning scheduler profits from the use of fixed priority criteria to build the BAT, such as RMS or DMS, in order to reduce the run-time scheduling overhead.

C. Previous Relevant Work on Response-Time Analysis

The most common temporal analysis for the periodic traffic in WorldFIP is solely on the verification that the traffic temporal constraints are met in the BAT. This can be done at the same time as the table is built and, thus, it is possible to consider the exact response time of each transaction. Response time can be defined, in this context, as the time between the periodic instant when a transaction becomes ready and when it is fully processed.

Concerning the aperiodic transfers, and due to their dynamic nature, a worst case response-time-based analysis is required whenever aperiodic variables are to be used to support real-time applications. In the literature it is possible to find several studies concerning the worst case response time to aperiodic requests in WorldFIP networks. Vasques and Juanole [25] derive an upper bound to the worst case response time for the aperiodic requests that includes all the traffic concerning the periodic transfers during the whole MC, plus the time required by all other aperiodic requests that can be issued during that period of time. This is overly pessimistic for many situations, particularly for medium to low network utilization. In such situations the ECs contain relatively large aperiodic windows that may allow processing the aperiodic requests, even when considered in the worst case, well before the end of the MC. This is noted by Pedro and Burns [17], who propose a less pessimistic analysis based on a lower bound to the aperiodic window of each EC. It is, then, considered that each EC has an aperiodic window with a fixed duration, equal to its lower bound, an assumption that, nevertheless, is still pessimistic. Moreover, all periodic transactions are considered to have the same duration, as well as all the aperiodic ones. Tovar and Vasques [23] propose a different analysis taking into consideration the existing BAT to account for the exact size of the periodic and aperiodic windows in each EC. This solution is the least pessimistic of the three. However, it highly depends on the particular way the BAT was built. For it to work properly, the BAT must be built considering all the periodic exchanges in phase (i.e., synchronous phasing) so that the critical instant is well defined. If the BAT was built using a nonsynchronous phasing among several periodic exchanges, then the critical instant becomes unknown unless an extensive search is carried out.

The analysis presented in this paper also considers the exact width of the periodic window in each EC, and consequently of the aperiodic window, in order to quantify how much aperiodic traffic can be processed. However, it allows circumventing the previous problem by basing itself not on the BAT but on the description of the variables' update requirements. Hence, despite a possible use of nonsynchronous phasing in the BAT, the analysis still considers the synchronous phasing, always resulting in an upper bound to the response time to aperiodic requests. Furthermore, this analysis does not require the pre-existence of a BAT and, thus, it can be used to analyze the schedulability of the aperiodic traffic before building the BAT, at an earlier design

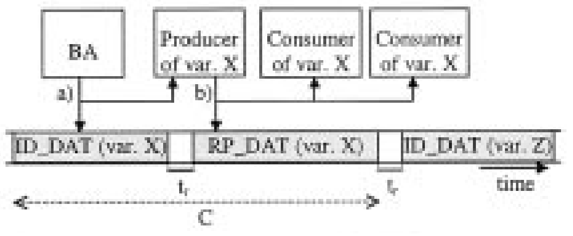

Fig. 3. Periodic exchanges.

phase. Moreover, it also has the advantage of being applicable to situations based on a dynamic BAT (e.g., planning scheduler). Finally, the following analysis can be applied both to periodic and aperiodic traffic, thus resulting in an integrated schedulability assessment for WorldFIP applications.

III. Analysis of Basic Exchanges in WorldFIP

The basic communication service for real-time applications is the periodic exchange of identified variables. The variables used by each node, either to consume or produce data values, are defined statically at pre-run time according to the application requirements. At each node, the protocol data link layer (DLL) manages a set of buffers holding the values of the exchanged variables. These buffers are locally available to the application software through application layer (AL) services.

The contents of the DLL buffers in the consumer nodes are automatically updated through a network service called *buffer transfer*. Each buffer transfer corresponds to a network transaction and includes an identification frame (ID_DAT) sent by the BA identifying the variable to be produced (*a)* in Fig. 3) and a response frame (RP_DAT) sent by the producer node and containing the respective updated value (*b)* in Fig. 3). When receiving the response frame, every consumer node will overwrite the respective DLL buffer with the variable's new value.

The duration of each transaction can be evaluated using expression (1), where tx_{rate} is the transmission rate expressed in bits per second, $\text{len}()$ is the bit length of a given frame, and t_r is the turnaround time in seconds

$$C = \frac{\text{len(ID\_DAT)} + \text{len(RP\_DAT)}}{tx\_\text{rate}} + 2 \times t_r. \tag{1}$$

Notice that while each ID_DAT frame is always 61 bits long, the RP_DAT frame is 61 bits long plus the data bits (2 bytes are used to code the type of the variable being transferred). The turnaround time is the time interval between the transmission of two consecutive frames, being defined by the protocol to be within 10–70 t_{mac} long, where t_{mac} is the time to transmit a physical symbol. This parameter is configured at system startup according to the propagation delays and the node's latency, having a considerable impact on the protocol efficiency.

Concerning the aperiodic buffer transfers, the situation is more complex. The BA processes them only after processing the periodic traffic in each EC, in the respective aperiodic window according to whether it is an urgent or normal request, or a message transfer request (Fig. 2). Without loss of generality, only urgent aperiodic requests will be considered. The aperiodic buffer transfers take place in three steps.

1) When processing the BAT schedule, the BA broadcasts an ID_DAT frame concerning a given periodic variable, say identifier X. The producer of variable X, that has a pending aperiodic request, responds with an RP_DAT frame, setting an aperiodic request bit in the control field of its response frame (step 1). The BA notices this bit and stores variable X in a queue of requests. There are, in fact, two queues, one for each aperiodic traffic priority level: urgent and normal.
2) In the aperiodic window, the BA uses an identification request frame (ID_RQ) to ask the producer of the X identifier to transmit its list of pending aperiodic requests (step 2). The producer of X responds with a RP_RQ frame (list of identifiers, e.g., identifiers Y and W). This list of identifiers is placed in another BA's queue, the *ongoing aperiodic queue*.
3) Finally, the BA processes the aperiodic transfers stored in the ongoing aperiodic queue (step 3). For each of these transfers, the BA uses the same mechanism as for periodic buffer transfers, i.e., ID_DAT followed by RP_DAT (Fig. 4).

From the three referred steps, it can be seen that an aperiodic transaction is made of several atomic transactions (ID frame/RP frame) that can be processed in several consecutive ECs. The first of these atomic transactions is for exchanging the identifiers list (step 2: ID_RQ/RP_RQ) followed by several buffer transfers in step 3, one for each variable requested in the list. Furthermore, step 1 implies that one aperiodic request might have to wait for a complete period of periodic variable X, until the BA becomes aware of the pending request. This period of time is defined as the *dead interval* [25] and is directly related to the shortest period among the periodic variables produced in the respective node (Fig. 4). This interval must be accounted for in the derivation of the worst case response time to an aperiodic transfer request.

IV. Computational Model and Worst Case Scenario

A computational model for both periodic and aperiodic exchanges in WorldFIP networks can now be established. Building upon this computational model, a worst case scenario for both periodic and aperiodic transactions will be defined.

A. Periodic Transfers

Consider a set Vp of Np process variables that must be broadcast periodically. Variable $v_i \in Vp$ is characterized by $v_i = v_i(C_i, D_i, P_i, Pr_i)$ for $i = 1, \ldots, Np$, where C_i is the duration of the respective transactions as in expression (1), D_i the relative deadline, P_i the period, and Pr_i the associated static priority. Note that, T2 being the EC duration, $\forall_{i=1,\ldots,Np} P_i = k * \text{T2}$, $k \geq 1$, and $C_i < \text{T2}$. Also, consider that the set of variables is ordered by decreasing priority, i.e., $\forall_{i,j=1,\ldots,Np}\ i > j => Pr_i \leq Pr_j$, and that inserted idle time is used to prevent transactions from crossing the EC boundary, or the periodic window if shorter than the EC.

Furthermore, the following analysis considers synchronized bus operation, only, in the sense that the activation of any transaction is always synchronous with the start of an EC. However,

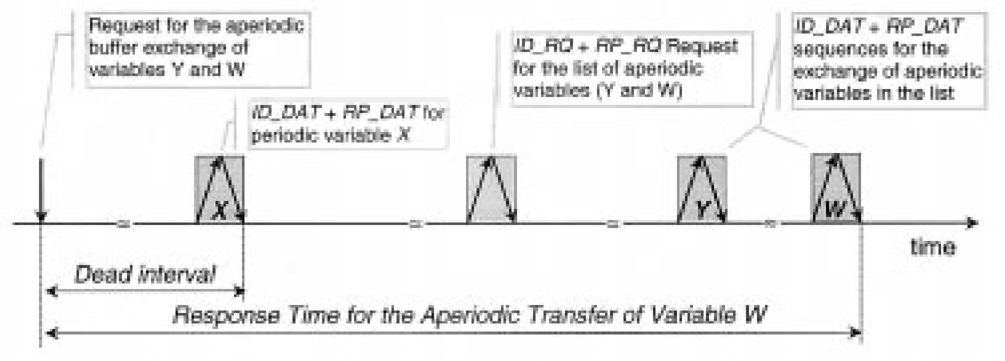
Request for the aperiodic buffer exchange of variables Y and W
ID_DAT + RP_DAT for periodic variable X
ID_RQ + RP_RQ Request for the list of aperiodic variables (Y and W)
ID_DAT + RP_DAT sequences for the exchange of aperiodic variables in the list
X
Y
W
time
Dead interval
Response Time for the Aperiodic Transfer of Variable W

Fig. 4. Elementary atomic transactions involved in the processing of an aperiodic request.

due to interference from higher priority variables, a transaction of a given variable v_i will have to wait for a certain amount of time to be effectively processed. The time interval from activation to complete processing is defined as the transaction response time. In general, this time may vary from activation to activation, causing communication jitter. For the purpose of the schedulability analysis, it is important to know which conditions maximize the response time and to determine its worst case value, $R\mathrm{wc}_i$ or, at least, an upper bound R_i to this value.

Similarly to what has been shown in [14], the worst case situation occurs when all the respective transactions become ready at the same instant, the critical instant, as long as a fixed-priority scheduling discipline is used.

B. Aperiodic Transfers

Consider a set Va of Na aperiodic process variables. The number of different nodes that request urgent aperiodic transfers is given by Ka. Each aperiodic variable $av_i \in Va$ is characterized by $av_i = av_i(k, \sigma_i^k, Ca_i, Da_i, Pra_i)$ for $i = 1, \ldots, Na$, where k is the node that requests the transfer of variable av_i and σ_i^k is the associated dead interval, Ca_i is the duration of the respective aperiodic buffer transfer, also given by expression (1), Da_i is its relative deadline, and Pra_i the associated priority: urgent or normal.

The time interval between the reception of an aperiodic transfer request for variable av_i by a given node, and the complete processing of the respective transaction is defined as the respective response time. An upper bound for the worst case response time will be represented by Ra_i.

Although the protocol does not consider relative deadlines (Da_i), some restrictions must be enforced to support a schedulability analysis for this sort of traffic. In particular, a lower bounded inter-arrival time must be considered for each aperiodic variable, in order to bound its interference on the response time of the other variables. In practice, the deadline Da_i will be used as such minimum inter-arrival time. Furthermore, the protocol considers that an aperiodic transfer for the same variable may be requested by any node, i.e., the respective producer, one of the respective consumers or by a third party node. However, this directly collides with an enforcement of the minimum inter-arrival interval. For example, two different nodes could then make two simultaneous requests for the same aperiodic variable that could be processed by the BA one shortly after the other. Hence, it will be considered that all the aperiodic requests for the same variable come from the same node, and that the application will not violate the minimum inter-arrival time between requests for the same variable.

There is, yet, another relevant feature concerning aperiodic requests in WorldFIP. The BA does not allow redundancy within the requests queue, i.e., any request coming from a given node is discarded whenever a previous request from the same node is still pending in the queue. Thus, if there are Ka nodes that can issue aperiodic requests then, the maximum number of requests in the queue is given by Ka.

Finally, in what concerns the aperiodic traffic, the protocol does not insert idle time to enforce a strict regularity of the EC duration. The EC end is signaled by the expiration of the T2 timer and, thus, if an atomic transaction is in progress, it will be carried out to completion, delaying the effective start of the next EC and causing release jitter to the periodic transfers. This jitter is bounded by the longest atomic transaction among aperiodic buffer transfers $(Ca_{i=1,\ldots,Na})$ and list requests $(Cl_{i=1,\ldots,Ka})$ and it must be accounted for when calculating the upper bound for the response time of periodic transactions (2). The padding frames transmitted by the BA when there is no traffic to transfer can also cause a similar jitter in the EC start, but of smaller magnitude and, thus, it will not be considered in this paper

$$R_i = R\mathrm{wc}_i + \max_{k=1,\ldots,Na, j=1,\ldots,Ka} (Ca_k, Cl_j). \tag{2}$$

C. Worst Case Situation for Aperiodic Requests

The determination of the worst case response time for the aperiodic transfer requests is reasonably more complex than for the periodic transfers. In this case, the worst case occurs when the following three conditions take place simultaneously (for clarity reasons, just the urgent queue is considered).

1) The request is placed in the BA's urgent queue only σ_i^k after it has been submitted by the application software in the requesting node k (maximum dead interval).
2) The arriving request is placed in the Kath position in the urgent aperiodic requests queue, which is served by the BA in a FCFS order. These Ka requests refer to all the Na aperiodic variables, i.e., each node requests all the aperiodic variables that it may request. Therefore, the total response time must include the transfer of both the

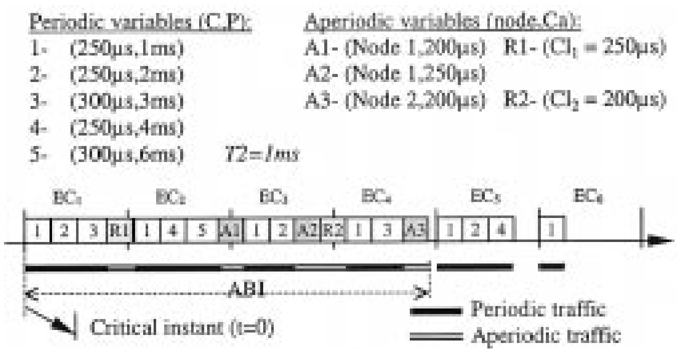

Fig. 5. Worst case response to aperiodic requests.

Ka list requests plus the Na aperiodic buffer transfers, which must be processed in the EC's aperiodic windows.

3) The request arrives at the BA at the beginning of the EC, where all periodic transactions become simultaneously ready (critical instant). This EC corresponds to the start of a sequence of ECs with the largest periodic traffic load.

The time interval between the critical instant and the end of the processing of all the aperiodic requests in the worst case situation is defined as the *aperiodic busy interval* (ABI) (Fig. 5). The order by which the list request exchanges and the aperiodic buffer transfers take place within the ABI is irrelevant for the analysis because of the FCFS behavior of the BA queue used to handle the aperiodic requests (2nd condition above). The worst case response time to an aperiodic request is then given by expression (3)

$$Ra_i = \sigma_i^k + \text{duration(ABI)}. \tag{3}$$

The duration of the list request exchanges, Cl_k, can also be obtained from expression (1) replacing the ID_DAT/RP_DAT frames with ID_RQ/RP_RQ frames. Moreover, notice that *len*(ID_RQ) $=$ 61 bits and *len*(RP_RQ) $= 45 + 16^*$ nid bits, where nid is the number of identifiers in the list.

D. Absolute Upper Bound for the Dead Interval

The dead interval that can affect the aperiodic requests of variable av_i, issued by a node k, can be upper bounded by σ_i^k, which refers to the shortest period among the periodic variables produced by that node. Constraining k to be fixed for each variable, i.e., each aperiodic variable is always requested from the same node, the absolute upper bound for the dead interval σ_i^k of aperiodic variable av_i is given by (4)

$$\sigma_i^k = P_j + R_j \text{ and } v_j : P_j = \min_{v_l \text{produced by} k} (P_l). \tag{4}$$

The addition of R_j to P_j upper bounds the interval between two consecutive exchanges of variable v_j.

V. Timeline Method: An Integrated Analysis

A. Building the Timeline for Periodic Exchanges

The timeline method has been presented by the authors in [4] for a general computing model with nonpreemption, elementary cycles, and inserted idle time to avoid blocking in the beginning of each cycle. This model fits the one being considered in this paper for the periodic traffic (Section IV-A) and, thus, the timeline analysis can be directly applied to it [3]. However, WorldFIP does not enforce idle-time insertion for the aperiodic traffic and, thus, the timeline method as presented initially must be adapted whenever aperiodic traffic is also considered.

Basically, the timeline method consists in drawing the traffic timeline starting from the critical instant up to the point in time where all the variables have been broadcast at least once. This method allows accounting precisely for the inserted idle time in each cycle, while other traditional approaches to response-time-based analysis do not [21].

Fig. 6 shows the algorithm to execute the timeline analysis, where k is the index to the Np periodic variables, n the index to each EC starting from the critical instant, Load_n is the communication load accumulator for the nth EC, and $\delta_{k,n}$ is a Boolean function that becomes 1 whenever a transaction of variable v_k is ready in the nth EC and 0 otherwise. Line 1 contains the critical instant condition, considering that the transactions of all variables are ready in the beginning of the 1st EC. Line 6 inserts idle time whenever a given transaction does not fit in the current EC. Line 9 calculates the worst case response times for all variables, Rwc_i $i = 1, \ldots, Np$, not considering the jitter induced by the aperiodic traffic.

This algorithm can also be used when the periodic window (P1) is limited to an interval smaller than the EC duration $(T_{\text{P1}} < \text{T2})$ (Fig. 2). In this case, it suffices to replace T2 by T_{P1} in line 6. If, upon termination, there is at least one Rwc_i that has not been evaluated, then at least one deadline has been violated. When Rwc_i has been evaluated for all Np periodic variables, the upper bounds $R_{i=1,\ldots,Np}$ can be readily obtained using expression (2). Then, it is possible to formulate the following sufficient schedulability condition.

```
1.  **for** (k=1;k≤N_p;k++){Rwc_k=0; δ_{k,1}=1;}
2.  **for** (n=1;(n ≤ ⌈D_{Np}/T2⌉ **and** Rwc_{Np}=0);n++){
3.      Load_n=0;
4.      **for** (k=1;k≤N_p;k++){
5.          δ_{k,n+1}= δ_{k,n};
6.          **if** (Load_n + δ_{k,n}*C_k <= T2) {
7.              Load_n = Load_n + δ_{k,n}*C_k;
8.              δ_{k,n+1}=0;
9.              **if** (Rwc_k=0) Rwc_k=(n-1)*T2+Load_n;
10.             }
11.         **if** (n **mod** P_k/T2=0) δ_{k,n+1}=1;
12.         }
13.     }
```

Fig. 6. Algorithm for the timeline analysis.

The set of periodic variables
$Vp \equiv \{v_i,\ i = 1, \ldots, Np\}$ *is schedulable*
if $\forall_{i=1,\ldots,Np}\ R_i \le D_i$

The proof of this condition is obvious. If all the response time upper bounds are smaller than the respective deadlines, then the real response times also are and, thus, the set is schedulable.

```
1.  for (k=1;k≤$N_p$;k++){$Rwc_k$=0; $\delta_{k,1}$=1;}
2.  for (k=1;k≤$K_a$+$N_a$;k++){$\delta a_{k,1}$=1;}
3.  lenABI=0; $Load_1$=0; $Loadi_1$=0;
4.  for (n=1;(n≤⌈maxD/T2⌉ and (lenABI=0 or $Rwc_{Np}$=0);n++){
5.      for (k=1;k≤$N_p$;k++){
6.          $\delta_{k,n+1}$= $\delta_{k,n}$;
7.          if ($Load_n$ + $\delta_{k,n}$*$C_k$ <= T2){
8.              $Load_n$ = $Load_n$ + $\delta_{k,n}$*$C_k$;
9.              $\delta_{k,n+1}$=0;
10.             if ($Rwc_k$=0) {$Rwc_k$=(n-1)*T2+$Load_n$;}
11.             }
12.         if (n mod $P_k$/T2=0) {$\delta_{k,n+1}$=1;}
13.         }
14.     for (k=1;k≤$K_a$+$N_a$;k++){
15.         $\delta a_{k,n+1}$= $\delta a_{k,n}$;
16.         if ($Load_n$ + $Loadi_n$ < T2) {
17.             if (k≤$K_a$) {$Load_n$=$Load_n$+$\delta a_{k,n}$*$Cl_k$;}
18.             else {$Load_n$=$Load_n$+$\delta a_{k,n}$*$Ca_{k-Ka}$;}
19.             $\delta a_{k,n+1}$=0;
20.             if (k=$K_a$+$N_a$) {lenABI=(n-1)*T2+$Load_n$+$Loadi_n$;}
21.             }
22.         }
23.     $Loadi_{n+1}$= $Loadi_n$+$Load_n$-T2; if ($Loadi_{n+1}$<0) $Loadi_{n+1}$=0;
24.     $Load_{n+1}$=0;
25.     }
```

Fig. 7. Enlarging the timeline analysis to include aperiodic transactions.

B. Extending the Timeline to Include Aperiodic Requests

The same timeline method explained above can simultaneously be used to determine the length of the ABI for aperiodic transactions. Then, expressions (3) and (4) can be used to derive an upper bound of the response time to a request for variable av_i (Ra_i).

The duration of the ABI is determined by the total time spent transmitting all atomic transactions associated to the aperiodic traffic. Notice that there are Ka list requests (one from each requester node) and Na buffer transfers, and that the respective transactions take $Cl_{i=1,\ldots,Ka}$ and $Ca_{i=1,\ldots,Na}$. The algorithm presented in Fig. 7 enlarges the scope of the one presented in Fig. 6 and combines the calculation of both the worst case response times for the periodic transactions and the duration of the ABI.

Basically, the outer cycle (lines 4–25) that goes through the ECs one by one starting from the critical instant contains now two cycles, one to place periodic transactions in the EC accounting for inserted idle time (lines 5–13) and another one to use the remaining time in each EC, without inserted idle time, to process aperiodic related transactions (lines 14–22). Since the EC boundary can now be crossed by ongoing aperiodic related transactions, each EC may start with a certain communication load already used up by the last transaction of the previous EC. This is accounted for with $\mathrm{Load}i_n$. MaxD in line 4 stands for the longest deadline among periodic and aperiodic variables.

Whenever the maximum duration of the periodic window (Fig. 2) is smaller than the EC duration $(T_{\mathrm{P1}} < \mathrm{T2})$, T2 must be replaced by T_{P1} in line 7. Similarly, whenever the aperiodic window is constrained to finish before the EC $(T_{\mathrm{P3}} < \mathrm{T2})$, then T2 must be replaced by T_{P3} in line 16.

Such as for the set of periodic variables, $Ra_{i=1,\ldots,Na}$ allows formulating the following sufficient schedulability condition for the aperiodic set Va.

The set of aperiodic variables
$Va \equiv \{av_i,\ \ i=1,\ldots,Na\}$*is schedulable*
if $\forall_{i=1,\ldots,Na}\ \ Ra_i \leq Da_i$

The proof of this condition is, again, obvious. As the response time to any aperiodic request cannot be larger than the respective upper bound $(Ra_{i=1,\ldots,Na})$, if these upper bounds are lower than the respective deadlines, then the set is schedulable.

C. Complexity and Practical Application

The time complexity of the basic algorithm in Fig. 6 is $O(N_{\mathrm{EC}} * R_M)$ where N_{EC} is the maximum number of transactions that may fit within one EC (bounded by a constant) and R_M is $\max_{i=1,\ldots,Np}(R\mathrm{wc}_i)$ (approximately proportional to $\Sigma_{i=1}^{Np}\lceil R_M/P_i\rceil$). Hence, the time complexity depends on the scanning periods of the variables. However, for most applications it is likely that R_M is smaller than a large subset of periods and thus the resulting complexity will be $O(Np)$. When this is not verified, the time complexity will grow to $O(Np^2)$. This trend has been experimentally verified with randomly generated sets of variables.

Concerning the practical application of this analysis, despite the fact that it can also be used offline to verify *a priori* the schedulability of static sets of variables (with less pessimism than other proposed analyses), its main advantage is that it is well adapted to be executed online. Therefore, this analysis can

TABLE I
VARIABLES SET DERIVED FROM THE PSA BENCHMARK

Variable ID	Producer node	Period (ms)	Data (bytes)	Max # of bits (id_dat+rp_dat)	C (µs)	Rwc (µs)	R (µs)
1	Engine controller	1	6	170	210	210	396
2	Wheel angle sensor	2	1	130	170	380	566
3	Engine controller	3	1	130	170	550	736
4	AGB	2	1	130	170	720	906
5	Device x	4	3	146	186	906	1092
6	Device x	6	3	146	186	1396	1582
7	Device x	4	2	138	178	1574	1760
8	Bodywork sensor	8	3	146	186	1760	1946
9	Device y	6	2	138	178	1938	2124
10	Engine controller	16	5	162	202	2752	2938
11	AGB	10	3	146	186	2938	3124
12	Device x	16	1	130	170	3550	3736

Variable ID	Requester node	Cl (µs)	Dead int (µs)	Data (bytes)	Max bits (id_dat+rp_dat)	Da (ms)	Ca (µs)	Ra (µs)
A1	Engine controller	162	1210	3	146	10	186	7074
A2	AGB	178	2720	1	130	12	170	8584
A3	AGB		2720	1	130	15	170	8584
A4	Bodywork sensor	178	9760	3	146	20	186	15624
A5	Bodywork sensor		9760	2	138	20	178	15624

Duration of the ABI = 5864µs

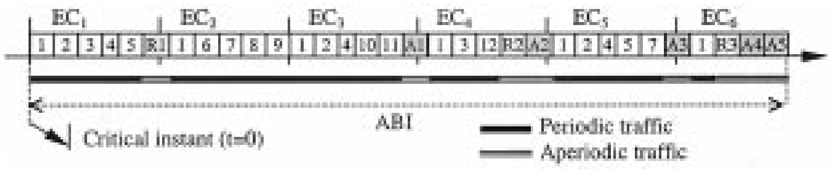

Fig. 8. Timeline for the set of variables in Table I.

be used in the scope of an online admission control, enabling the support of flexible communication requirements, such as in the case of networks using the planning scheduler [2].

Two different levels of flexibility can be offered by the system. In one case, the system accepts change requests issued by a human operator, such as adding new variables to the BAT, remove variables from the BAT or change the properties of variables in the BAT. In this case, a response time in the order of a few seconds is acceptable and the time available to perform the timeline analysis is relatively relaxed. Experiments with sets of 16 variables and bus utilizations around 50% were analyzed on an 8051 microcontroller in less than 100 ms.

A more demanding level of flexibility is needed when the change requests come from the application as response to environment changes, e.g., increasing the scanning rate of a variable upon detection of a given event. In this case, the system reaction to the change request may be constrained to a few milliseconds. Therefore, the analysis can hardly be executed in low processing power microprocessors. A solution that has been considered is the use of a field-programmable-gate-array (FPGA)-based coprocessor that schedules traffic online, accepts changes to the set of variables, and executes the timeline schedulability analysis. A prototype of such coprocessor has been presented in [15]. Preliminary performance figures show that the coprocessor is able to test the schedulability of sets with 32 variables in less than one EC. Despite having been designed for the Flexible Time-Triggered communication protocol Controller Area Network (FTT-CAN) networks, this coprocessor can also be directly used in WorldFIP networks to support dynamic traffic scheduling.

VI. NUMERICAL EXAMPLE

Consider the set of communication requirements expressed in Table I. These were taken from the benchmark supplied by PSA Peugeot Citroen and presented in [16]. The periods, however, have been altered in order to better suit the purposes of the example. The deadlines are equal to the periods and the IDs express the priorities in reverse order, i.e., $\mathrm{ID} = 1$ has highest priority and $\mathrm{ID} = 12$ has lowest. Also, five aperiodic variables have been added, which can be requested by one of three nodes, as expressed in Table I.

Considering a transmission rate of 1 Mbit/s and a turnaround time of $20t_{\mathrm{mac}}$, i.e., $t_r = 20\ \mu\mathrm{s}$, expression (1) can be used directly to determine the duration of each buffer transfer, resulting in $C_{i=1,\ldots,12}$ for the periodic variables and $Ca_{i=A1,\ldots,A5}$ for aperiodic ones. The duration of the list request exchanges $Cl_{i=1,\ldots,3}$ is determined considering the number of aperiodic variables requested by each node. The EC duration (T2) is 1 ms and it can be fully dedicated to the periodic traffic $(T_{\mathrm{P1}} = \mathrm{T2})$.

The timeline method (Section V) allows obtaining $R\mathrm{wc}_{i=1,\ldots,12}$ as well as the duration of the ABI (Fig. 8). Then, through expression (2), $R_{i=1,\ldots,12}$ is calculated. Since $R_i \leq P_i\ \forall_{i=1,\ldots,12}$, then the set of periodic variables is schedulable. In what concerns the aperiodic variables, the upper bound to the dead interval in each node ($\sigma_i^k\ k = 1, \ldots, 3$) is calculated according to expression (4) and, finally, the upper bounds to the response times ($Ra_{i=A1,\ldots,A5}$) are determined using (3). Since $Ra_i \leq Da_i\ \forall_{i=A1,\ldots,A5}$, the set of aperiodic variables, given the set of periodic ones, is also schedulable.

It is interesting to remark on the relative magnitude of the response times for the periodic traffic and for the aperiodic one. The larger upper bounds of the latter one are due to two cumulative factors: the inefficient mechanism used to handle aperiodic transactions, which requires at least one periodic transaction and one list request before the aperiodic variable is broadcast, and the worst case behavior of an FCFS queue, which forces it to consider each request as the last one in a full queue.

VII. Conclusions

This paper has discussed the schedulability analysis of the real-time traffic in WorldFIP networks. In particular, it proposes using the timeline method to evaluate the response time of variable exchanges, based on the orderly evaluation of the elapsed bus time. This timeline methodology is suitable to situations using inserted idle time, as is the case with the periodic traffic in WorldFIP when operating in synchronous mode. Furthermore, the analysis is based on the variables scanning requirements, considering fixed-priorities scheduling and all variables in phase, and not on the BAT itself. Thus, if a different phasing is used in the BAT, e.g., to level the periodic load in the ECs or to reduce jitter, the worst case properties of the analysis are still respected.

In this paper, the timeline method was adapted in order to allow deriving upper bounds for the worst case response times of both periodic and aperiodic transactions in WorldFIP networks, in an integrated approach. The method is implemented by means of a bounded cyclic process that has relatively low CPU requirements. It is thus possible to use it online as part of an admission control protocol to guarantee that dynamic changes to the communication requirements do not jeopardize the system timeliness. This feature is of particular interest when dynamic mechanisms are used in the WorldFIP protocol, such as the planning scheduler, allowing for online changes to the set of periodic variables. Using a specialized hardware coprocessor to perform the traffic scheduling and schedulability analysis can further improve the dynamic behavior of WorldFIP networks.

References

[1] *FIP Bus for Exchange of Information Between Transmitters, Actuators and Programmable Controllers*, French Standards NF C46601–C46607, 1989–1992.

[2] L. Almeida, R. Pasadas, and J. A. Fonseca, “Using a planning scheduler to improve flexibility in real-time fieldbus networks,” *Control Eng. Practice*, vol. 7, pp. 101–108, 1999.

[3] L. Almeida and J. A. Fonseca, “Schedulability analysis in the FIP fieldbus accounting for inserted idle-time,” presented at the Euromicro Conf. Real-Time Systems RTS’99 (Work-in-Progress Session), York, U.K., June 1999.

[4] ——, “Analysis of a simple model for nonpreemptive blocking-free scheduling,” in *Proc. Euromicro Conf. Real-Time Systems RTS’01*, Delft, The Netherlands, June, 2001, pp. 233–240.

[5] N. C. Audsley, A. Burns, and A. J. Wellings, “Deadline monotonic scheduling theory and application,” *Control Eng. Practice*, vol. 1, pp. 71–78, 1993.

[6] “General purpose field communication system,” CENELEC, Brussels, Belgium, EN 50170, 1996.

[7] J. Dworzecki, “Ordonnancement déterministe des tâches périodiques en présence de contraintes temporelles et de sucessions,” in *Proc. Real-Time Embedded Systems RTS’98*, Paris, France, 1998, pp. 235–251.

[8] D. Galara and J. P. Thomesse, “Groupe de réflexion FIP. Proposition d’un système de transmission série multiplexée pour les échanges d’information entre des capteurs, des actionneurs et des automates réflexes,” Min. l’Industrie Recherche, Paris, France, May 1984.

[9] S. H. Hong, “Scheduling algorithm of data sampling times in the integrated communication and control systems,” *IEEE Trans. Contr. Syst. Technol.*, vol. 3, pp. 225–230, June 1995.

[10] *Industrial Automation Systems—Manufacturing Message Specification (MMS)*, ISO 9506, 1990.

[11] M. Joseph and P. Pandya, “Finding response times in a real-time system,” *Comp. J.*, vol. 2, no. 5, pp. 390–395, 1986.

[12] Y. S. Kim, S. Jeong, and W. H. Kwon, “A pre-run-time scheduling method for distributed real-time systems in a FIP environment,” *Control Eng. Practice*, vol. 6, pp. 103–109, 1998.

[13] S. Kumaran and J. D. Decotignie, “Multicycle operations in a fieldbus: Application layer implications,” in *Proc. IEEE IECON’89*, Philadelphia, PA, 1989, pp. 531–536.

[14] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[15] E. Martins and J. A. Fonseca, “Improving flexibility and responsiveness in FTT-CAN with a scheduling coprocessor,” in *Proc. FET’01—4th IFAC Conf. Fieldbus Systems and Their Applications*, Nancy, France, Nov. 2001, pp. 61–67.

[16] N. Navet and Y.-Q. Song, “Design of reliable real-time applications distributed over CAN (controller area network),” in *Proc. IFAC Symp. Information Control in Manufacturing, INCOM’98*, Nancy, France, June 1998, pp. 391–396.

[17] P. Pedro and A. Burns, “Worst case response time analysis of hard real-time sporadic traffic in FIP networks,” in *Proc. 9th Euromicro Workshop Real-Time Systems*, June 1997, pp. 3–10.

[18] P. Raja and G. Noubir, “Static and dynamic polling mechanisms for fieldbus networks,” *Oper. Syst. Rev.*, vol. 27, no. 3, pp. 34–45, 1993.

[19] J.-P. Thomesse, “Time and industrial local area networks,” in *Proc. COMPEURO’93*, Paris, France, 1993, pp. 365–374.

[20] ——, “A review of the fieldbuses,” *Annu. Rev. Control*, vol. 22, pp. 35–45, 1998.

[21] K. Tindell *et al.*, “Analysis of hard real-time communications,” *J. Real-Time Syst.*, vol. 9, pp. 147–171, 1995.

[22] E. Tovar, F. Vasques, and L. M. Pinho, “Engineering real-time applications with WorldFIP: Analysis and tools,” in *Proc. SICICA 2000*, Buenos Aires, Argentina, 2000, pp. 297–302.

[23] E. Tovar and F. Vasques, “Contribution for the worst-case response time analysis of real-time sporadic traffic in WorldFIP networks,” presented at the Euromicro Conference on Real-Time Systems RTS’99 (Work-in-Progress Session), York, U.K., June 1999.

[24] ——, “Distributed computing for the factory-floor: A real-time approach using WorldFIP networks,” *Comput. Ind.*, vol. 44, pp. 11–31, 2001.

[25] F. Vasques and G. Juanole, “Pre-run-time schedulability analysis in fieldbus networks,” in *Proc. IEEE IECON’94*, 1994, pp. 1200–1204.

Luís Almeida (S’86–M’89) received the degree in electronics and telecommunications engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, Aveiro, Portugal, in 1988 and 1999, respectively.

He has been an Assistant Professor in the Department of Electronics and Telecommunications, University of Aveiro, since 1999. He is also a Senior Researcher with the IEETA research unit at the university. Formerly, he was a Design Engineer with a company producing digital telecommunications equipment. His research interests lie in the fields of real-time networks for distributed industrial/embedded systems and navigation control for mobile robots. He is a coauthor of a national patent concerning a fieldbus communication system, and he regularly participates in scientific events in both of his fields of interest.

Eduardo Tovar (M'98) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1990, 1995, and 1999, respectively.

Currently, he is a Professor of Industrial Computer Engineering in the Computer Engineering Department, Polytechnic Institute of Porto (ISEP-IPP), where he is also engaged in research on real-time distributed systems and factory communications. Since 1991, he has authored or coauthored more than 40 technical papers in the areas of factory communications, real-time systems, and industrial computer engineering.

José Alberto G. Fonseca (M'00) was born in Aveiro, Portugal, in 1957. He graduated in electronics and telecommunications engineering and received the Ph.D. degree in electrical engineering from the University of Aveiro, Aveiro, Portugal, in 1980 and 1992, respectively.

He has been an Associate Professor in the Electronics and Telecommunications Department of the University of Aveiro, Aveiro, Portugal, since 2000. His current research interests are embedded systems, distributed systems, and industrial communications. Presently, he is the Co-ordinator of two research projects funded by the Portuguese government in the fields of distributed embedded systems and environment monitoring systems. Since 1992, he has authored or coauthored more than 40 technical papers and has recently submitted two patents for fieldbus-based systems.

Francisco Vasques (M'99) received the B.Sc. degree in electrical engineering from the University of Porto, Porto, Portugal, in 1987, and the M.Sc. and Ph.D. degrees in computer science from the LAAS-CNRS, Toulouse, France, in 1992 and 1996, respectively.

He is currently an Assistant Professor in the Department of Mechanical Engineering, University of Porto, where he teaches the real-time systems course within the Electrical and Computer Engineering M.Sc. Degree Program. He has served as a program committee member for several conferences in the area of real-time systems and factory communications. Since 1991, he has authored or coauthored more than 50 technical papers in the areas of real-time systems, factory communications, and real-time systems architectures.

Dr. Vasques was the Program Co-Chairman of the 2000 IEEE Workshop on Factory Communications.