Meeting Real-Time Constraints in CAN

Salvatore Cavalieri

***Abstract*—This paper deals with the problem of information flow scheduling in a communication system based on CAN ISO IS-11898 physical medium access mechanism. It mainly features a bus access arbitration protocol based on a priority assigned to each message to be transmitted; if two or more messages are transmitted at the same time by different communication nodes, only the message with the highest priority continues to be transmitted, the other being stopped. In real-time applications, messages are relevant to process variables which must be transmitted within strict time constraints; according to the CAN ISO IS-11898 bus arbitration protocol, respect of real-time constraints depends on the priority assigned to each variable. The aim of the paper is to propose a procedure for dynamic assignment of priorities to variables to be transmitted, in such a way the relevant real-time requirements are fulfilled. Although many other approaches can be found in literature, the proposal is original as it is based on standard full CAN communication stacks.**

***Index Terms*—CAL CiA, CAN, CANOpen, fieldbus, real-time, scheduling.**

I. Introduction

AMONG fieldbus systems [1], [2], CAN ISO IS-11 898 is one of the oldest and most used communication system in industrial applications. CAN IS-11 898 features a bus topology and a carrier sensor multiple access/collision avoidance (CSMA/CA) mechanism based on the presence of a dominant bit (i.e., bit zero) [3]. Information on the bus is sent in fixed format messages of limited length. An Identifier field is univocally associated to each message to be transmitted. The identifier does not indicate the destination of the message, but is related to the meaning of the data, so that all nodes in the network are able to decide by message filtering whether the data is to be acted upon by them or not. As a consequence of the concept of message filtering, any number of nodes can receive and simultaneously act upon the same message. The identifier is a sequence of 11 or 29 bits, depending on the version of the CAN (CAN Specification 2.0 part A or B, respectively [3], [4]).

The message identifier defines a static message priority during bus access; when the bus is free any communication node connected to the bus may start to transmit a message. If two or more nodes start transmitting messages at the same time, the bus access conflict is resolved by an arbitration using the identifier. During arbitration, each transmitter compares the level of the bit currently transmitted with the level that is monitored on the bus. If these levels are equal, the communication node may continue to send. When a "recessive" level (i.e., a bit 1) is sent and a "dominant" level (i.e., a bit 0) is monitored by a node, this last one has lost arbitration and must withdraw without sending one more bit. Transmission of the message by the communication node that has lost arbitration is simply delayed; for this reason, the mechanism of arbitration guarantees that no information is lost.

Manuscript received December 13, 2004; revised January 20, 2005.

The author is with the University of Catania, Faculty of Engineering, Department of Computer and Telecommunications Engineering, Catania, Italy (e-mail: Salvatore.Cavalieri@diit.unict.it).

Digital Object Identifier 10.1109/TII.2005.844429

CAN is used in a wide range of industrial applications. Most of them are classified as real-time applications whose correctness depends not only on the logical results of computing, but also on the time at which the results are produced [5]. In real-time applications, producer and consumer processes exchange periodic variables, featuring a period and a deadline. A period is the time interval between to subsequent production of a variable; a deadline is the maximum amount of time interval starting from each production, within which the variable produced must be received by the consumer process; sometimed the deadline coincides with the period.

In a bus-based communication system like CAN, each variable produced must be properly scheduled to access the network and to be consumed within its deadline. On account of what said about arbitration mechanism in CAN ISO IS-11898, respect of the deadline for each variable transmitted depends on the identifier assigned to the CAN message containing each of its values.

CAN ISO IS-11 898 standard does not specify any rule for the assignment of identifier to message. For this reason, a lot of research activities have been carried on during these last years, aiming to define scheduling approach able to fix message identifiers according to the transmission requirements and constraints of each variable to be transmitted. The reader can refer to [6]–[9] for examples of scheduling approaches in CAN ISO IS-11 898.

Nowadays, several communication systems featuring a full communication stack based on CAN ISO IS-11 898 are available. They mainly define an application layer above the ISO IS-11 898 Data Link and Physical layers, making available powerful sets of services for the communication between control applications. Generally, they also offer network management services, including those to assign, store and retrieve the CAN identifier. Use of these systems clearly lead to more powerful applications and to a reduction of the cost of software development. Examples of these systems are the CAL CiA [10], CANOpen [11], [12], DeviceNet [13], and SDS [14].

The aim of the paper is to propose an approach aimed to assign identifiers to messages taking into account real-time requirements of the variable carried by each of them; one of the main features of the proposed approach is its integration in a full standard communication stack based on CAN ISO IS-11 898. The proposal is original, as literature presents proposal for assignment of CAN identifiers based only on a CAN ISO IS-11 898 Physical and Data Link layers, as was stated before. The advantage of the proposed approach is that to allow both real-time exchange of information (due to the identifier assignment) and availability of all the powerful communication services offered by a full communication stack. In order to achieve this, the proposal must be compliant with the services available at that each layer of the full communication stack, and in particular with those (if present) in charge to handle the CAN identifiers.

In the paper, a procedure for the assignment of CAN identifiers will be presented considering two of the most known communication systems based on CAN ISO IS-11 898: CAL CiA [10] and CANOpen [11], [12]. Both have been defined by CAN in Automation, International Users and Manufacturers Group (CiA), in 1996. The second one became an European standard in 2001, as it was included by CENELEC in the EN 50325, together with DeviceNet and SDS.

The paper will clearly describe the CAN identifier assignment procedure and will demonstrate its full compliance with the CAL CiA and the CANOpen communication stacks. Finally, a performance evaluation will point out the capability of the proposed approach to respect real-time requirements better than those currently achieved by CAL CiA and CANOpen standards.

Table I summarizes the acronyms used in the paper, in order to help the reader in the understanding of the explanation of the proposal.

II. Procedure for the Assignment of CAN Identifiers

The procedure proposed in the paper for the assignment of identifiers to messages is based on a centralised policy. It is assumed that one of the communication nodes in a CAN acts as master and manages the assignment of the identifiers. The node responsible for this is called the schedule master (SM) and the other nodes are the schedule slaves (SSs). They are the communication nodes connected to sensors/actuators/controllers, so producing information flow featuring both real-time (RT) and non-real-time (NRT) constraints. As known, RT traffic may be further divided into hard real-time (HRT) and soft real-time (SRT) [15].

The proposed procedure features different behaviors according to the operational phase of the CAN communication system: system start-up and run-time.

At system start-up, the SM collects the list of the time requirements to be fulfilled, from each SS connected to the network. Then the SM runs a scheduling algorithm in order to assign the identifiers to each request and distributes to the SSs the identifiers so far found.

At run-time, when a new node is added to the network and/or a SS already present at start-up needs to obtain new identifiers, the SM has to collect the novel assignment requests and has to assign and distribute the identifiers to each requesting SS.

At system run-time, a re-assignment procedure may be also played by the SM, when one or more identifiers do not fulfil time requirements of the relevant time critical processes.

It is important to point out that the proposal is not linked to specific scheduling algorithms the SM has to run in order to assign the identifiers to the requesting SSs. Sections III and IV will give some details about the features that a scheduling algorithm must offer; every scheduling algorithm satisfying the constraints pointed out in those sections can be adopted for the SM.

TABLE I
Acronyms Used in the Paper

CAL	CAN Application Layer
CAN	Controller Area Network
CiA	CAN in Automation, International Users and Manufacturers Group
CMS	CAN-based Message Specification
COB	Communication Object
COB-ID	Communication Object Identifier
CSMA/CA	Carrier Sensor Multiple Access/Collision Avoidance
DBT	COB-ID Distributor
ISO	International Organisation for Standardisation
LMT	Layer Management
NMT	Network Management
PDO	Process Data Object
SE	Service Element
SM	Schedule Master
SS	Schedule Slave
SDO	Service Data Object
WCRT	Worst Case Response Time

III. Assignment of CAN Identifiers in CAL CiA

In this section, first a brief overview of the CAL CiA standard will be given, pointing out the main features regarding the assignment of identifiers. The overview will allow to highlight the limits of the current priority assignment offered by the standard CAL CiA and the advantage of the proposal here presented. Finally, the implementation of the assignment procedure described before, using services and procedure of the CAL CiA standard, will be clearly introduced. The description will allow to highlight the perfect compliance of the proposed procedure with the existing standard.

A. A Brief Overview of CAL CiA

CAL CiA standard is made up by three layers: Physical, Data Link, and Application [10]. Physical and Data Link layers are those defined in [3], [4]. The Application layer provides for services for the exchange of information between applications, to be used upon the CAN ISO IS-11 898 layers.

Communication functionalities offered by the CAL CiA standard, are logically divided over four services elements (SEs); distributed applications exchange information using services and procedure of these SEs. They are: CAN-based Message Specification (CMS) [16], [17], Network Management (NMT) [18], Distributor (DBT) [19], [20], and Layer Management (LMT) [21]. In this section, only CMS and DBT are described, as they are strictly linked to the implementation of the proposal.

CMS offers services for the exchange of information between applications, allowing each of them to model its behavior in the form of objects [each of which is called a communication object (COB)] and remote services on these objects.

Communication between COBs is realized using a client/server model. Each application needing to exchange information with other ones, has to define a COB specifying several items among which: the kind of information exchanged, the role (Client or Server) played by the application exchanging information, the length in bytes of the transmission, and the number of clients or servers involved in the transmission. A COB must be defined for each exchange of information performed by each application.

TABLE II
CMS SERVICES

CMS Service	Description
DomainDownload	It allows a client to send a set of large data to a server.
NotifyEvent	A Server notifies to client(s) that a particular event has occurred and supplies its value.

TABLE III
DBT SERVICES

DBT Service	Description
CreateUserDefinition	A node, whishing to transmit, asks the DBT Master to have assigned a COB-ID, invoking this service, passing several information, among which the COB Class (i.e. priority) requested and a COB Name. COB-ID assignment performed by the DBT Master has been explained in the text.
DeleteUserDefinition	A node may delete all its user definitions inside the COB Database, specifying its node number. It's not possible for a node to delete a single user definition.

Each COB features a priority, called COB-ID, which is directly mapped onto the CAN ISO identifier. In the CAL CiA standard, COB-IDs are grouped into eight priority classes, each called COB Class; COB Class 0 features the highest priority and COB Class 7 the lowest one. Considering a CAN using an 11-bit identifier, 1760 different COB-IDs are available, spread into the eight priority classes: from 1 to 220 (class 0), 221 to 440 (class 1), 441 to 660 (class 2), 661 to 880 (class 3), 881 to 1100 (class 4), 1101 to 1320 (class 5), 1321 to 1540 (class 6) and 1541 to 1760 (class 7).

CMS makes available services for the exchange between COBs. These services allow to define variables (of a certain type), to read/write variables, to transfer (download and upload) domains (i.e., blocks of large data), to define, notify, store, and read events. Table II summarizes some of services available, only those used in the proposal.

CAL CiA standard foresees a dynamic COB-ID assignment; the DBT SE is in charge to offer services and procedure for the assignment of COB-IDs. A particular entity called DBT Master plays this role, and maintains a COB DataBase, containing all the COB-IDs assigned to the applications.

Each application needing to exchange information has to request a COB-ID to the DBT Master; as the information exchange may involve two or more applications, all those applications wishing to be involved in a particular exchange of information (e.g., several applications need to receive the same variable produced by another application) have to receive the same COB-ID, in such a way the message filtering described in the introduction can be realized.

For this reason, the COB Database is structured into COB Definitions; a COB Definition is univocally identified inside the COB Database by its identifier, the COB-ID, and contains all the User Definitions created for each application exchanging information using that COB-ID. A User Definition contains, among others, the identifier of the requesting node and a COB Name. A COB Definition may also contain a Predefinition, to which a COB Name is associated. Predefinition allows dynamic assignment of COB-IDs based on a pre-assigned scheduling, as it will be described in the following.

A node, whishing to transmit, asks the DBT Master to have assigned a COB-ID. The node passes to the DBT master several information, among which the COB Class (i.e., priority) requested and a COB Name. The DBT Master looks for a COB Definition with a Predefinition containing the requested COB Name. If it exists, the COB-ID corresponding to the COB Definition, will be returned to the requesting node. If no predefinition exists with the specified COB Name, a COB Definition containing a User Definition with the same COB Name is looked for. If any, the relevant COB-ID is assigned to the node. If no User Definitions containing the same COB Name are found, the DBT Master will create a novel COB Definition with a COB-ID of the same COB Class requested. If there are no free COB-IDs corresponding to the priority requested, the DBT Master has to assign a COB-ID with a lower priority than the one requested.

DBT offers several services, among which Table III summarizes only those used in the proposal.

B. Main Limits of the CAL CiA Priority Assignment Procedure

The overview of the CAL CiA standard given in the previous section allowed to highlight that the assignment criteria adopted by the DBT Master is only based on two information received from the requesting node: the COB Class desired and the status of the COB Database (i.e., if free COB-IDs of the requesting class exist).

It is clear that this mechanism leads to a COB-ID assignment strictly linked to the order with which the nodes make their COB-ID assignment requests, as the DBT Master cannot take into account previous or next assignment requests, while performing each COB-ID assignment. For example, let us assume that at a certain instant a node makes a request for a certain COB Class, obtaining a COB-ID of the requested priority. Later, another node makes a request relevant to the same COB Class and related to a more critical time requirement (e.g., shorter deadline and period). The DBT Master may find no COB-IDs free inside the COB Class requested, or it can find a free COB-ID of the requested COB Class, but greater (so featuring a lower priority) than the COB-ID previously assigned. In these two cases, a more critical application will have assigned a priority lower than that assigned to a less critical process.

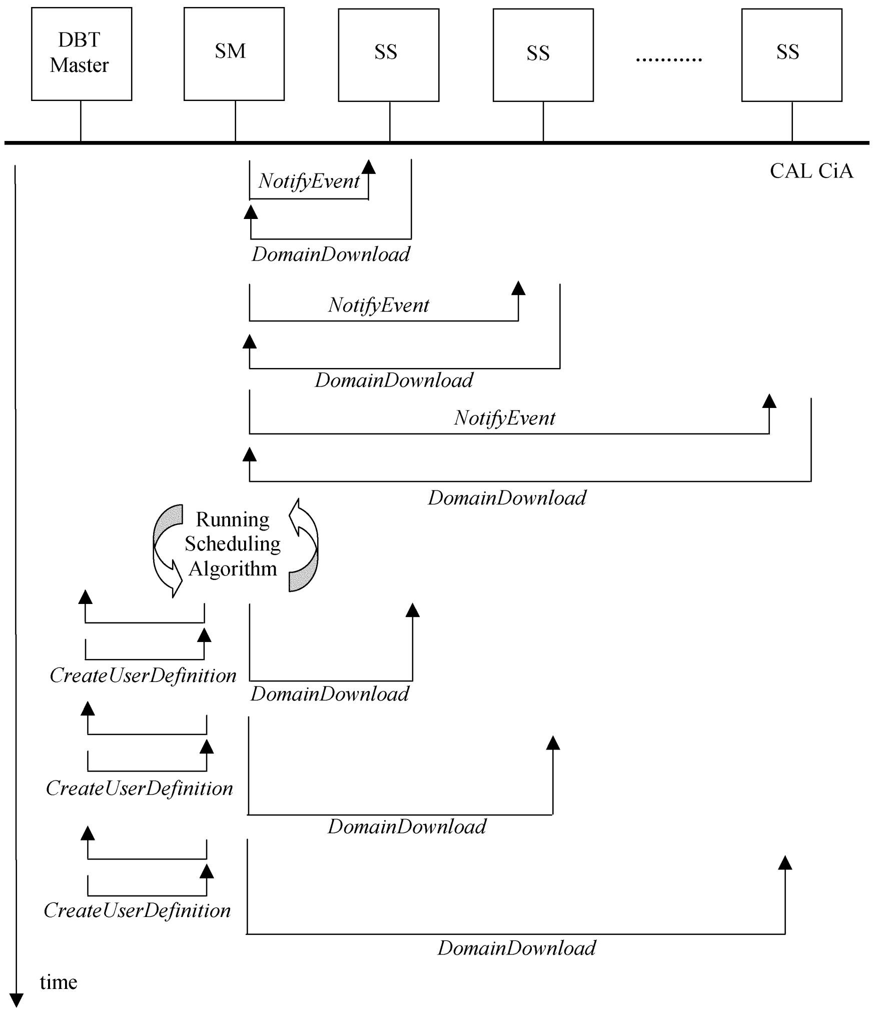

Fig. 1. Proposed COB-ID assignment at system start-up in CAL CiA.

The limits of the assignment criteria operated by the DBT Master explain the reason for a novel strategy of dynamic priority assignment for the CAL CiA. The proposed priority assignment procedure is based on a centralised scheduling of the assignment requests, which collects all the assignment requests and tries to take into account previous and next assignments while performing a single one, as explained in the following. In order to maintain the full compliance with the standard, the implementation of the proposed assignment procedure is perfectly integrated with CAL CiA CMS and DBT services and protocol (including all the activities carried on by the DBT Master), as shown in the following.

C. A Detailed Description of the Priority Assignment Procedure in CAL CiA

Description of the implementation of the COB-ID assignment procedure based on CAL CiA standard, will be split into three parts, referring to the actions required at system start-up and at run-time. Description of the main features of the scheduling algorithm that the SM has to run, will be also given for each of the three parts.

1) COB-ID Assignment at System Start-Up: At system start-up, it is assumed that no activities can be performed by SSs, until the SM has started a polling procedure. The SM polls all the nodes using the CMS *NotifyEvent* service, sent to each SSs. For each *NotifyEvent*, the SM waits for an answer from the relevant SS. If no answer is received within a time-window (e.g., because the node is not connected to the network), the SM sends another *NotifyEvent* to the next SS, if any. With this service the SM requires to each SS transmission of all its assignment requests, including deadline, period, and the RT type (HRT, SRT, or NRT).

If the SS is active and connected to the network reacts to the previous service by preparing and sending one or more COB-ID assignment requests, using the *DomainDownload* service and specifying, for each of them, the deadline, the period, and the RT type of the communication requirement.

The polling procedure at start-up ends when the SM has polled all the SSs in the network, having collected all the relevant COB-ID assignment requests.

The SM will run a suitable scheduling algorithm, in order to assign a COB Class to each request received (details of the scheduling algorithm will be given in the following). Then, for each assignment request, it will invoke the DBT *CreateUserDefinition* service specifying, among others, COB Name, COB Class (evaluated before by the scheduling algorithm), and Node-ID (i.e., the identifier of the node to which the assignment request refers). As explained in Section III-A, according to this service the DBT Master will assign a COB-ID and will send it to the requesting node (i.e., the SM) through a response to the previous *CreateUserDefinition*. Each COB-ID received from the DBT Master, will be distributed by the SM to the relevant node through the CMS *DomainDownload* service.

Fig. 1 shows all the information flow transmitted in the CAL CiA system, relevant to the start-up phase.

a) Scheduling algorithm for COB-ID assignment at start-up: The main aim of this algorithm is that to determine a COB Class for each assignment request received, as was stated before. The definition of the main features of this algorithm was also done taking into account the need to overcome the limits of the standard COB-ID assignment procedure, pointed out in Section III-B.

Once the SM has collected all the assignment requests at start-up, the scheduling algorithm has to do the following.

- Sort all the requests using the RT time (from the HRT to NRT type), as primary key; then, for each RT type, all the requests must be sorted according to deadline and period, respectively.
- Assign a COB Class to each assignment request. Due to the presence of only eight priority classes, a COB Class must be assigned to a group of assignment requests, starting with Class 0 assigned to the group of the most critical processes. The scheduling algorithm must avoid to empty each COB Class; in fact, if a COB Class becomes empty (i.e., no COB-ID are available inside it), no run-time assignment requests for that class can be satisfied in the future. In order to prevent this, the scheduling algorithm has to evaluate the average number of COB-ID assignment requests for each class, given by the total number of requests divided for the number of COB Classes, i.e., 8. Said $\mathrm{N_{COB}}$ this number, the scheduling algorithm will assign a COB Class to each group made up of $\mathrm{N_{COB}}$ priority assignment requests, starting from the group featuring the most critical requirements and assigning them Class 0, as stated.
- Request to the DBT Master a COB-ID at time, starting from the hardest RT request with the smallest deadline and period.
- Store each COB-ID assigned by the DBT Master inside a local scheduling table. This table will be used during the run-time operational phase, as described later.

The proposed scheduling algorithm is able to overcome the limits of the standard DBT-based COB-ID assignment. In fact, now the DBT Master is forced to assign the COB-ID starting from the most critical time requirements, avoiding that a more critical process has a COB-ID assigned featuring a priority lower than a less critical process. Further, assigning only $\mathrm{N_{COB}}$ COB-ID for each class, the probability that a COB Class requested has been previously emptied is reduced at run-time, when other priority assignment requests will be done, as explained in the following.

2) Run-Time COB-ID Assignment: After start-up phase, new transmission requirements may arise from both already existing SSs and SSs added to the network after start-up phase. A procedure to collect novel assignment requirements and to assign COB-IDs has been foreseen. It performs differently according if the SS is added to the network or it is already present at the start-up phase.

In the first case, as was stated before, it is assumed that no activities can be performed by a SS, until the SM has started a polling procedure. The SM periodically polls all the nodes not present at the start-up phase, using the CMS *NotifyEvent* service. For each *NotifyEvent* invoked, the SM waits for an answer from the relevant SS; if no answer is received within a time-window, the SM polls the next SS (in the list of SSs not present at start-up phase). As was done for the start-up phase (see Fig. 1), the *NotifyEvent* service is used by the SM to require, from each SS, the transmission of all its priority assignment requests. If the polled SS is active and connected to the network reacts to the previous service by preparing and sending one or more assignment requests, using the *DomainDownload* service.

If a SS already present in the network during the start-up phase, has the need to send to the SM one or more novel assignment requests, it will directly use a *DomainDonwload* service.

For each priority assignment request collected at run-time, the SM has to assign the proper COB Class. Then it has to request the COB-ID to the DBT Master using the *CreateUserDefinition* service. Finally, each COB-ID received from the DBT Master, will be distributed to the relevant SS, using the *DomainDownload* service.

b) Scheduling algorithm for COB-ID assignment at run-time: Scheduling is more critical at run-time, as new assignment requests must be satisfied taking into account all the assignments done before; this means that, for each novel assignment request, the COB Class value to be assigned must be very close to that relevant to already scheduled assignment requests with similar time requirements. For example, let us assume that the current assignment request to be scheduled features a HRT type, a deadline and a period of 8 ms. Further, let us suppose that the scheduling table presents a group of HRT requests with deadline ranging from 7 to 10 ms, with COB-IDs assigned belonging to COB Class 1. In this case, it is clear that the scheduling algorithm has to assign COB Class 1 to the current assignment request. But, if the group of HRT requests with deadline ranging from 7 to 10 ms, features COB Classes 0 and 1, the scheduling algorithm has to assign the highest priority COB Class (i.e., Class 0), in order to increase the probabilities to fulfil the time constraints of the current assignment request.

On the basis of what was stated, the scheduling algorithm running inside the SM has to:

- open the scheduling table;
- look for a group of already scheduled assignment requests featuring RT type, deadline and period values very close to that hold by the current assignment request. If this group features the same COB Class value, it must be assigned to the current request; on the other hand, if the group is made up by COB-IDs belonging to several COB Classes; the highest priority one must be assigned to the current request.

3) Run-Time COB-ID Re-Assignment: COB-ID re-assignment procedure is needed when one or more "unsatisfactory" COB-ID assignments occur. Definition of unsatisfactory assignment will be given in the following.

As seen before, the scheduling algorithm performed by the SM (both at start-up and run-time phases) aims to force the DBT Master to assign COB-IDs in such a way the highest priority COB-IDs (i.e., the COB-IDs with the lowest values) are assigned to the hardest RT processes with the shortest deadline.

It is clear that this assignment strategy cannot guarantee the respect of all the time requirements, i.e., generally it is not possible to state that all the deadlines of the processes will be certainly respected.

Literature presents several techniques to ascertain this. For example, in [6], a worst-case response time (WCRT) evaluation technique has been shown. Considering a CAN, WCRT refers to the worst case time interval starting from when a variable has been produced by a node until the message containing the variable has been successfully transmitted and received by another node of the same network. This time interval depends on the priority of the message identifier, as the transmission of the message occurs only when all higher priority messages have been transmitted. Annex A will describe this technique applied to the 11-bit identifier CAN, allowing to determine the WCRT for each message sent in CAN.

Considering each process in a CAN and considering the relevant deadline, it is clear that if the WCRT evaluated for the variables produced by the process is higher than its deadline, the relevant time requirements may be not satisfied, e.g., a deadline may be missed. But, it is also clear that it is impossible to foresee when a deadline will be missed and how many deadlines will be missed.

On the basis of what said, it is proposed that, for each COB-ID assigned, the SM will check by a WCRT-based analysis if the relevant time requirements could be satisfied. If one or more COB-IDs whose relevant deadline may be missed, are found, the SM will start a re-assignment procedure. The procedure is done for a COB-ID(s) at time, involving the SSs sharing a COB-ID to be re-assigned. When re-assignment for a particular COB-ID has been completed, re-assignment procedure for another COB-ID is started.

For each SS holding the COB-ID to be re-assigned, the SM will first notify the need to start a re-assignment procedure. This will be done using the CMS *NotifyEvent* service. On reception of the notification of the service, the SS will invoke the DBT service *DeleteUserDefinition*, in order to delete all the User Definitions contained in the COB Database concerning the note itself. This procedure is required by the DBT protocol, as it is not possible to delete a single user definition for a particular COB-ID, but only all the user definitions relevant to the node. After having deleted all its user definition, the SS will send again all its assignment requests to the SM. These include those relevant to the COB-ID to be re-assigned and those already existing, as they were deleted though the *DeleteUserDefinition* service call. The SM will collect all the requests for the SS and will run the run-time scheduling algorithm described above. In this case, the scheduling algorithm will assign the same COB Class for the COB-ID deleted that must not be re-assigned, and it will assign a higher priority COB Class for the COB-ID to be re-assigned. The SM will invoke the *CreateUserDefinition* to have the COB-IDs re-assigned. Once received the new assigned COB-IDs, the SM will distribute them to the SS. In the meantime, the re-assignment procedure has not completed for a specific node, the SS will continue to use the old COB-ID. As can be verified in the standard, this does not cause any problem. When the re-assignment procedure has been completed for a COB-ID, the SM will notify the conclusion of the procedure and will force the SSs sharing it to update the assigned COB-ID through the *NotifyEvent* service. From that moment, those SSs will use the new COB-ID.

As the re-assignment procedure is very time consuming, it cannot be realized on-line during COB-ID assignments at start-up or run-time phases. So, it was foreseen that periodically the SM evaluates all the previous assignments and activate the re-assignment procedure if one or more COB-IDs not able to respect the deadline of the process are found. The time interval for the periodic execution of the re-assignment procedure is a very critical parameter. A very short period leads to a high computational overload inside the SM and a very high bandwidth utilization; on the other hand a very long period may lead to a very high number of deadline missed.

It is important to point out that the proposed re-assignment mechanism cannot guarantee that the new COB-IDs assigned will led to the full respect of all the time constraints not respected before the re-assignment. Some of them could be missed also after the re-assignment procedure. This means that the SM must carefully evaluate when and if a re-assignment procedure must be executed. A forecast based on simulation and WCRT analysis can be used by the SM to take a decision about when execution of a re-assignment procedure is advantageous and leads to a better fulfilment of the RT constrains not respected.

IV. Assignment of Identifiers in CANOpen Standard

As was done before, first a brief overview of CANOpen system will be given. Then, implementation of the proposal based on CANOpen services and procedures will be described.

A. A Brief Overview of CANOpen

In CANOpen. a very important role is played by the Object Dictionary hold by each CANOpen device; it is a group of communication objects accessible via the network, using a 16-bit index. According to [11] and [12], communication in CANOpen is realized through two types of communication objects: a process data object (PDO) and a service data object (SDO).

The RT data transfer is performed by means of PDO on a producer/consumer(s) basis; PDO producer uses Transmit-PDO, while PDO consumer supports Receive-PDO. In CANOpen, for each device, it is possible to configure at most 512 Transmit and 512 Receive PDOs. Each of them corresponds to two entries in the device Object Dictionary: a Communication Parameter and a Mapping Parameter. Communication Parameter specifies, among others, the COB-ID used for the exchange of information; like CAL CiA, the COB-ID is directly mapped onto the CAN message identifier. Mapping Parameter specifies the real objects mapped to the PDO-based information flow, e.g., the variable exchanged. One of the PDO services allowing a producer to send data to consumer(s) is the *WritePDO* one, that will be used in the proposal here presented.

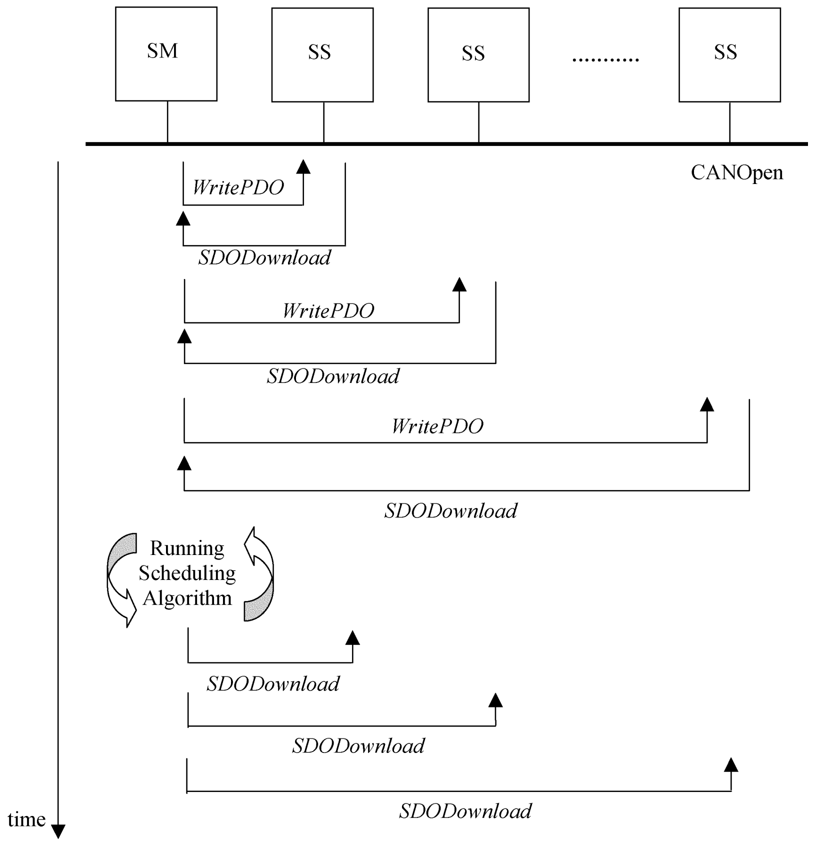

Fig. 2. Proposed COB-ID assignment at system start-up in CANOpen.

SDO is used to transfer larger data set, containing arbitrary large block of data; basically a SDO is transferred as a sequence of segments. Another difference with PDO is that SDO communication is based on a client/server model. Each SDO corresponds to an entry in the device Object Dictionary. According to the role of the node in the client/server communication model, two kinds of SDO entry are present in the Object Dictionary: the Server and the Client SDO Parameter: they are used by the responding and the requesting node, respectively. For each device, it is possible to configure at most 128 Server SDO Parameters and 128 Client SDO Parameters. Among other things, a Server/Client SDO Parameter specifies the COB-IDs used for the exchange of information. One of the services defined for the SDO-based communication is the *SDODownload*, that will be used in the proposal; through this service, the client sends data to the server.

Like CAL CiA, the COB-ID assignment in CANOpen network may be realized through the DBT services [10], [19], [20]. But unlike CAL CiA, it is important to point out that the DBT Management is not mandatory in the CANOpen standard [11]. In the case DBT Master was not used, COB-ID assignment is performed in a so-called "pre-operational" state (i.e., at system start-up, before system run-time), using SDO services.

B. Main Limits of the CANOpen Priority Assignment Procedure

As stated, DBT services to dynamically assign COB-IDs are not mandatory in CANOpen. If no DBT Master is present, the standard foresees only a static assignment at system start-up. On the other hand, if DBT Master is used, the limits relevant to the dynamic COB-ID assignment highlighted in Section III-B, arise.

Due to the availability to avoid to use the DBT-based services for the COB-ID assignment, here is proposed to overcome the limits of the DBT-based COB-ID dynamic assignment, implementing the proposed procedure for the assignment of CAN identifiers without taking into account the presence of a DBT Master, as shown in the following.

C. A Detailed Description of the Priority Assignment Procedure in CANOpen

As was done for the CAL CiA standard, description of the implementation of the procedure proposed for COB-ID assignment based on CANOpen standard, will be divided into three parts: system start-up and the actions performed at run-time (i.e., assignment and re-assignment of COB-IDs). The main features of the scheduling algorithm the SM has to run, will be pointed out for both the start-up and run-time phases.

1) COB-ID Assignment at System Start-Up: At the system start-up, the SM is the only node in the network enabled to start the communication; all the SSs are in a stand-by condition, listening for any command sent by the SM and reacting to his command as explained in the following.

The SM polls all the nodes, invoking for each of them the PDO *WritePDO* service, in order to compel each SS to transmit COB-ID assignment request(s). After the SM has polled a particular node, it waits until a time-out expires. If the SM does not receive any request from the SS, it will pass to poll the next node in the network, if any. On the other hand, if the SS is connected to the network, reacts to the previous service notification by preparing and sending one or more COB-ID assignment requests, using the SDO *SDODownload* service.

Unlike what done for the CAL CiA, in the implementation based on CANOpen it was assumed to not consider the DBT Master, as stated before. Due to this choice, once the SM has collected all the requests coming from the SSs connected to the network, it will run a suitable scheduling algorithm, in order to evaluate the COB-IDs to be assigned to each SS, and will distribute them to each SS using the *SDODownload* service.

Fig. 2 shows the start-up phase of the proposal in the CANOpen system.

c) Scheduling algorithm for COB-ID assignment at start-up: Due to the lack of the DBT Master, the COB-ID assignment must be done by the SM, on the basis of the assignment requests (i.e., RT type, deadline, and period) received during the start-up phase. In order to try to satisfy all the assignment requests, the scheduling algorithm must:

- sort all the COB-ID assignment requests, according to the real-type time (from the HRT to the NRT one); for each RT type, all the requests must be sorted according to deadline and period, respectively;
- assign a COB-ID to each assignment request, once they have been sorted. The COB-ID assignment must be done assigning the highest priority COB-IDs starting from the most critical assignment requests. As was stated in Section III-C1a, it is very important to avoid emptying each COB Class, as if COB-IDs are no more available inside a COB Class, no run-time assignment requests for that class can be satisfied in the future. In order to prevent this, the scheduling algorithm will evaluate the N_{COB} number [described in Section III-C1a)] and will assign only the first N_{COB} COB-IDs of each COB Class to the assignment requests.

As was done for the implementation based on CAL CiA, it is assumed that the SM will maintain a scheduling table containing all the COB-IDs assigned by the scheduling algorithm. This table will be used during the run-time operational phase, as will be described later.

2) Run-Time COB-ID Assignment: As it was assumed that a new added node cannot start any activities until an explicit command from the SM of the network is received, the procedure here proposed foresees that at run-time the SM periodically polls all the nodes not included in its list of active nodes. This list has been built at system start-up during the polling procedure described before. Polling and collection of COB-ID assignment requests occur as described for the start-up phase; the PDO *WritePDO* service is used by the SM to poll the nodes not included in the list of active nodes and the *SDODownload* service is used to collect COB-ID assignment requests from SSs.

Let us now consider another scenario, featured by a SS present in the network at system start-up and requesting at run-time new COB-ID assignments. The SS will notify this event to the SM by a PDO *WritePDO* service, making aware the SM of the COB-ID assignment need featured by the particular SS. After this notification, the SS prepares and sends an explicit COB-ID assignment request, using the *SDODownload* service.

In both cases, the SM has to assign the COB-IDs to the requests received and has to distribute the assigned COB-IDs to the SSs; distribution of COB-ID is done through the SDO *SDODownload* service, like the start-up phase.

d) Scheduling algorithm for COB-ID assignment at run-time: Assignment of a COB-ID at run-time may be not so easy like in the start-up phase, as was stated for the CAL CiA implementation. In order to understand better the difficulties that the scheduling algorithm running inside the SM may encounter, let us assume that COB-ID values belonging to the range [900–1000] have been already assigned to HRT process with deadline ranging from 7 to 10 ms; further, let us assume that a new HRT process featured by a deadline of 8 ms requests a COB-ID assignment. If COB-ID 970 has been assigned to a HRT process with deadline 9, and COB-ID 950 has been assigned to a HRT process with deadline 7, it is clear that a COB-ID value ranging from 950 to 970 must be assigned to the novel request. But, if no COB-IDs are available in the previous interval]950–970[, a COB-ID value outside it, must be assigned. In order to better guarantee that the RT requirements of the current request will be met, a higher priority COB-ID value should be chosen (i.e., lower than 950).

On the basis of what said, for each assignment request received at run-time, the scheduling algorithm should:

- open the scheduling table;
- look for the first already scheduled assignment request featuring a lower RT type or deadline or period than that relevant to the current one. If any, the scheduling algorithm will assign the first COB-ID value featuring a priority higher than that found. If no request featuring a lower RT type or deadline or period has been found, a COB-ID featuring a priority lower than the lowest priority found in the scheduling table, will be assigned.

3) Run-Time COB-ID Re-Assignment: As stated in Section III-C.3, COB-ID re-assignment procedure is needed when assignments do not guarantee the respect of a set of RT requirements.

As was done for the CAL CiA, here it is proposed that the SM periodically will check for each COB-ID assigned, if the relevant time requirements could be satisfied; this may be done by a WCRT analysis (see Annex A). If one or more COB-IDs, whose relevant deadlines may be missed, have been found, the SM will start a re-assignment procedure for these COB-IDs. For each COB-ID to be re-assigned, the Schedule Master has to:

- re-assign the “unsatisfactory” COB-ID to a higher priority value, that is able to the respect of the deadlines. A WCRT analysis may be used also in this case to find the most suitable value;
- send a new COB-ID to the SSs holding each re-assigned COB-ID. This is done using the PDO *WritePDO* service.

V. Performance Evaluation

Assessment of the proposed priority assignment procedures based on CAL CiA and CANOpen has been carried on taking

TABLE IV
SCENARIO 1

Variables	Period (msec)	Deadline (msec)	Real-Time Type	Size (bytes)
Start-up				
50 (25 for each SS)	50	50	Hard	5
200 (100 for each SS)	200	200	Hard	5
200 (100 for each SS)	300	300	Hard	5
200 (100 for each SS)	500	500	Soft	5
200 (100 for each SS)	600	600	Soft	5
Run-time				
30 (15 for each SS)	100	40	Hard	5

into account that their main aim is the respect of the RT constraints (e.g., deadline) of each variable exchanged in a CAN. For this reason, assessment has been finalised to verify the capability of the proposal to respect the RT constraints of each process present in a CAN.

Introduction pointed out that literature presents a lot of paper about scheduling solutions in communication systems based on the CAN ISO IS-11 898 Data Link and Physical layer protocols and adopting a proprietary solution at the Application layer. As the proposal here presented is based on standard Application layers, a comparison between the scheduling solutions proposed in this paper and those available in literature has not considered. In fact, in the case of some kind of difference in the assessment, it would be very difficult to attribute its absolute dependence on the scheduling solutions, due to the clear influence of the different number and features of communication layers on which they are based.

Comparison of the proposed priority assignment procedure with that based on DBT services present in CAL CiA and in CANOpen, has been assumed to be more meaningful, as the solutions compared are based on the same communication stack.

A. Methodology and Hypotheses Adopted

Estelle [22]–[25] has been used to model the COB-ID assignment procedure performed by the DBT Master (according to CAL CiA and CANOpen standards), the proposed COB-ID assignment procedure based on CAL CiA and the proposed COB-ID assignment procedure based on CANOpen. The use of Estelle allowed also to validate the COB-ID assignment procedures presented in this paper.

These models were used to run simulations, using Estelle tools [26]–[30]. Simulations were aimed to realize a performance evaluation, considering different scenarios, each featuring particular values of the following parameters:

- CAN bandwidth.
- Time constraints for each variable: RT type (HRT, SRT, NRT), deadline, period, and size of the variable (in bytes).
- Number of variables featuring the same time constraints.
- Number of variables whose assignment requests arise at the start-up and run-time phases of the proposed assignment procedures. This means, for example, that if 100 variables featuring the same time constraints are considered, different scenarios could be obtained varying the percentage of these variables assigned during start-up and run-time phases (for example 40 variables assigned at start-up and 60 at run-time phase). As seen before, these two phases do not exist in the standard DBT-based assignment procedure; in order to compare the performance evaluation results, it was assumed that the order and timing of the assignment requests for the variables taken into account in each scenario, is exactly the same in the standard DBT-based assignment procedure and in the proposed assignment procedure on CAL CiA and CANOpen. So coming back to the previous example, featuring 40 variables assigned at start-up and 60 at run-time phase, the requests of assignment of these variables arise at the same times, considering the standard DBT-based network and in the proposed procedure running in CAL CiA and CANOpen.
- The number of nodes producing information to be exchanged was not considered a very important parameter for the simulation. The most important one was the number variables to be assigned, as was stated before. So it was assumed to have a very small network with only two nodes producing variables; for each scenario, it was assumed to split the total workload in the two nodes, in equal parts. It is clear that the total number of nodes present in the network may be not the same; in the standard DBT-based network, only two nodes (producing the same traffic, as was stated before) and the DBT Master have been considered. In the assignment procedure based on CAL CiA, two SSs (producing information), a SM (performing only its role) and the DBT Master have been considered. Finally, the assignment procedure based on CANOpen features only two SSs and a SM.

For each scenario, three simulations were performed, one for each of the three Estelle-based models seen before. Each simulation was aimed to obtain the COB-ID assignment realized according to the standard DBT-based COB-ID assignment, to the proposed assignment procedure based on the CAL CiA standard and that based on CANOpen standard. On the basis of the COB-IDs assigned to each variable, the WCRT analysis presented in Annex A was applied; the WCRT value of each variable has been compared with the relevant deadline, in order to check if this last has been respected.

In the following, some of the most significant results of the performance evaluation carried on will be presented.

B. Scenario 1

Table IV summarizes the communication scenario taken into consideration. As can be seen, 880 variables have been considered, featuring HRT and SRT constraints. Size, period and dead-

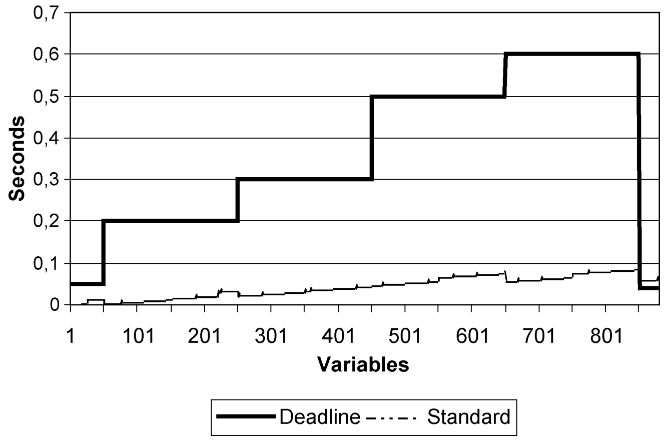

Fig. 3. Scenario 1: WCRT versus deadline in the standard DBT-based assignment.

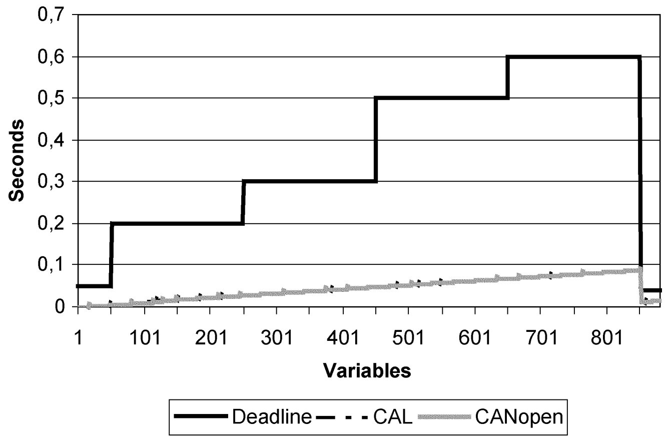

Fig. 4. Scenario 1: WCRT versus deadline in the proposed assignment procedure running in CAL CiA and CANOpen.

TABLE V
SCENARIO 2

Variables	Period (msec)	Deadline (msec)	Real-Time Type	Size (bytes)
		Start-up		
160	40	40	Hard	5
160	50	50	Hard	5
160	70	70	Hard	5
160	100	100	Soft	5
160	120	120	Soft	5
160	150	150	Non-Real-Time	5
160	200	200	Non-Real-Time	5
		Run-time		
50	30	30	Hard	5
60	60	60	Hard	5
60	80	80	Hard	5

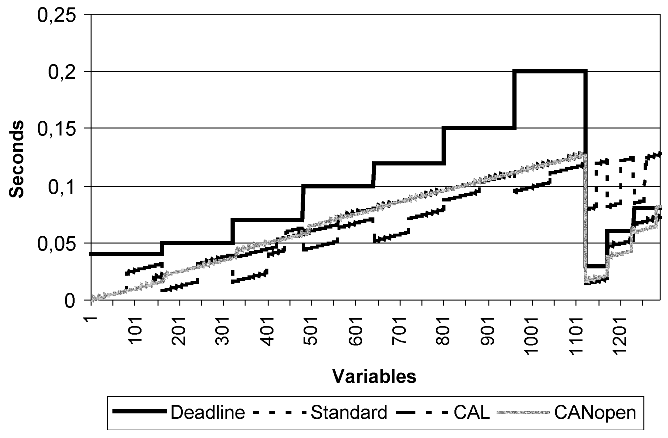

Fig. 5. Scenario 2: WCRT versus deadline in the standard DBT-based assignment and in the proposed assignment procedure running in CAL CiA and CANOpen.

line of the variables are shown in the table. A communication bandwidth of 1 Mbps has been considered.

The table also shows the variables whose assignment request is made at system start-up and at run-time phase, for the proposed procedures running in a CAL CiA and CANOpen. Considering the standard DBT-based assignment, the table specifies only the order with which the assignment requests of the variables arise.

Fig. 3 shows the WCRT relevant to the standard DBT-based COB-ID assignment (Standard curve) versus the deadline of each variable (Deadline curve). Fig. 4 compares the deadline of each variable with the WCRT values relevant to the proposals here presented running in a CAL CiA network (CAL curve) and in a CANOpen network (CANOpen curve), respectively.

Comparing Figs. 3 and 4 it is possible to point out that the standard DBT-based COB-ID assignment fails to respect deadlines of the last 30 variables shown in Table IV. As can be seen from Fig. 4, deadlines of these variables are fully respected in the proposal running in CAL CiA and CANOpen systems. The DBT-based priority assignment mechanism mainly fails on account of the order of the priority assignment requests, given in Table IV. The presence of the first 450 HRT variables forces the standard DBT Master to assign all the COB-IDs belonging to the higher priority COB Classes. So when the 30 HRT variables featuring a deadline of 40 ms must be assigned, no more COB-IDs of high priority are available, and the DBT Master must assign lower priority COB-IDs, causing the missing deadlines shown in Fig. 3. This is avoided in the proposal here presented due to the centralised scheduling performed by SM, which prevents to empty each COB Class, using the mechanism based on the evaluation of N_{COB} value.

C. Scenario 2

Table V summarizes the communication scenario taken into consideration. Again a communication bandwidth of 1 Mbps has been considered.

Fig. 5 shows the WCRT versus the deadline of each variable, considering the standard COB-ID assignment, and the proposal here presented running in a CAL CiA network and in a CANOpen network. It allows to highlight that the standard COB-ID assignment based on DBT fails to respect deadlines of the last 170 HRT variables shown in Table V (featuring deadlines of 30, 60, and 80 ms, respectively). As can be seen from the figure, deadlines of these variables are fully respected in the proposal running in CAL CiA and CANOpens systems. These results confirm all the remarks for Scenario 1.

TABLE VI
SCENARIO 3

Variables	Period (msec)	Deadline (msec)	Real-Time Type	Size (bytes)
Start-up				
200	100	50	Hard	5
300	100	100	Soft	5
200	200	200	Non-Real-Time	5
Run-time				
100	100	40	Hard	5
100	100	80	Hard	5
100	150	150	Soft	5
100	200	180	Soft	5
100	220	220	Non-Real-Time	5
100	300	300	Non-Real-Time	5

TABLE VII
SCENARIO 4

Variables	Period (msec)	Deadline (msec)	Real-Time Type	Size (bytes)
Start-up				
200	100	50	Hard	3
300	100	100	Soft	10
200	200	200	Non-Real-Time	30
Run-time				
100	100	40	Hard	3
100	100	80	Hard	3
100	150	150	Soft	10
100	200	180	Soft	10
100	220	220	Non-Real-Time	30
100	300	300	Non-Real-Time	30

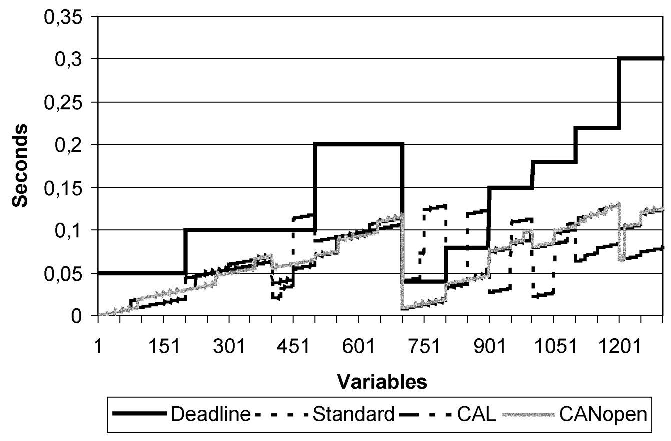

Fig. 6. Scenario 3: WCRT versus deadline in the standard DBT-based assignment and in the proposed assignment procedure running in CAL CiA and CANOpen.

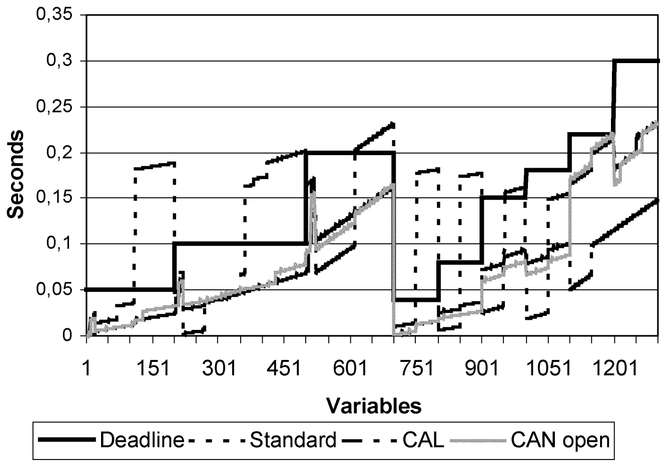

Fig. 7. Scenario 4: WCRT versus deadline in the standard DBT-based assignment and in the proposed assignment procedure running in CAL CiA and CANOpen.

D. Scenario 3

Table VI summarizes the communication scenario taken into consideration. As was done for the other scenarios, a communication bandwidth of 1 Mbps has been considered.

Fig. 6 shows the WCRT versus the deadline of each variable. Again, it is possible to point out that the standard DBT-based COB-ID assignment fails to respect deadlines when the last SRT variables, featuring a 100-ms deadline, are requested to be assigned at start-up phase. Further, the DBT-based standard assignment fails when the HRT variables (featuring deadlines of 40 and 80 ms) are introduced in the run-time phase. This behavior is again due to the limit of the DBT Master policy, as pointed out several times in the paper.

E. Scenario 4

The workload considered in this scenario is the same used in Scenario 3, changing the size of the variables, as shown in Table VII.

As can be seen from Fig. 7, the results shown for the previous scenarios are all confirmed. This scenario allows to highlights much more the limits of the standard DBT—based assignment protocol present in CAL CiA and CANOpen.

VI. FINAL REMARKS

The paper has presented a protocol for dynamic priority assignment in two CAN-based standards: CAL CiA and CANOpen. The performance evaluation carried on during the research here presented, allowed to point out that the standard priority assignment strategy foreseen in CAL CiA and CANOpen may fail, missing deadlines of critical processes. The proposal allows achieving a better respect of RT constraints.

The proposal is relevant from the scientific point of view, as it represents the only solution known in literature, perfectly compliant with standard Application Layers, as shown in the paper. Other approaches to improve RT features of CAN system are present in literature, but they mainly adopt a proprietary application and/or user layer upon the standard Data Link and Physical CAN layers. Introduction pointed out the advantages in proposing a RT solution using a standard application layer.

The impact of the proposal is not limited to academic environment. The adoption of standard application layer services and protocol allows a very easy and fast software implementation using existing commercially available CAL CiA and/or CANOpen libraries; in this way, many industrial applications may take advantage from a development of software realising the protocol here presented, using CAN software and hardware already existing in the industrial environment.

VII. Annex A

The priority-based access mechanism featured by the CAN system, allows using the theory of the schedulability analyis in a single processor system, in order to verify if each deadline is respected. In this context, the WCRT of a message $\boldsymbol{m}$, from the moment it is produced to the instant at which it transmission has been concluded is given by

$$\mathrm{R_m} = \mathrm{I_m} + \mathrm{C_m} \tag{1}$$

where $\mathrm{I_m}$ is the waiting time due to the transmission of the higher priority messages or to the currently transmission of lower priority messages (i.e., transmission started before the message m has been produced) and $\mathrm{C_m}$ is the time needed to transmit message m. It is clear that for each message m, the set of higher priority messages is made up by all the messages featuring a higher priority COB-ID and waiting to be transmitted.

Said $\mathrm{B_m}$ the blocking factor as the time needed to complete the current transmission of the lower priority message, it is possible to write

$$\mathrm{I_m} = \mathrm{B_m} + \sum_{\forall \mathrm{j} \in \mathrm{hp(m)}} \left(\left\lceil \frac{\mathrm{I_m} + \tau_{\mathrm{bit}}}{\mathrm{T_j}} \right\rceil \cdot \mathrm{C_j} \right) \tag{2}$$

where $\mathrm{hp}(m)$ is the set of message featuring a higher priority respect message m, τ_{bit} is the bit time, $\mathrm{T_j}$ is the period of the generic message j in the set $\mathrm{hp}(m)$, $\mathrm{C_j}$ is the time needed to transmit message j, and

$$\mathrm{B_m} = \max_{\forall \mathrm{k} \in \mathrm{lp(m)}} \{0, \mathrm{C_k}\}$$

with $\mathrm{lp}(m)$ being the set of lower priority messages respect m.

As CAN foresees a 47-bit overhead for each message (11 bit identifier CAN type A), and among these 47 bits, 34 undergo a stuffing process realized inserting a stuff bit every 5 bits (see [3] and [4]), it is possible to define

$$\mathrm{C_m} = \left(\left\lfloor \frac{34 + 8 \cdot \mathrm{nb_m}}{5} \right\rfloor + 47 + 8 \cdot \mathrm{nb_m} \right) \cdot \tau_{\mathrm{bit}} \tag{3}$$

where $\mathrm{nb_m}$ is the number of bytes of message m.

References

[1] J. D. Decotignie and P. Pleinevaux, “Time critical communication networks: Field buses,” *IEEE Network*, vol. 2, no. 3, pp. 55–63, May 1988.
[2] ——, “A survey on industrial communication networks,” *Annales Telecommun.*, vol. 48, pp. 9–10.
[3] ISO, Road Vehicle—Interchange of Digital Information—Controller Area Network (CAN) for High Speed Communication, 1993.
[4] R. Bosch, CAN Specification Version 2.0, 1991.
[5] K. Avind, K. Ramamritham, and J. A. Stankovic, “A local area network architecture for communication in distributed real-time systems,” *Real-Time Syst.*, vol. 3, no. 2, 1991.
[6] E. Tovar, “Supporting Real-Time Communications With Standard Factory-Floor Networks,” Ph.D. dissertation, Univ. Porto, Porto, Portugal, 1999.
[7] S. Ouni and F. Kamoun, “An efficient protocol for hard real-time communication on controller area network,” in *Proc. 6th Int. CAN Conf.’99*, 1999, pp. 4–16.
[8] K. W. Tindell, H. Hamsson, and A. J. Wellings, “Calculating controller area network (CAN) messages response times,” *Control Eng. Practice*, vol. 3, no. 8, 1995.
[9] ——, “Analysing real-time communications: controller area network (CAN),” in *Proc. Real-Time Systems Symp.*, 1994, pp. 259–263.
[10] CiA, CAN Application Layer for Industrial Applications: CAN in the OSI Reference Model, 1996.
[11] Cenelec, Industrial Communications Subsystem Based on ISO 11 898 (CAN) for Controller-Device Interfaces. Part 4: CANOpen, 2001.
[12] CiA, CAL-Based Communication Profile for Industrial Systems-CANOpen. Version 3.0, 1996.
[13] Cenelec, Industrial Communications Subsystem Based on ISO 11 898 (CAN) for Controller-Device Interfaces. Part 2: DeviceNet, 2001.
[14] ——, Industrial Communications Subsystem Based on ISO 11 898 (CAN) for Controller-Device Interfaces. Part 3: Smart Distributed System-SDS, 2001.
[15] J. Xu and D. L. Parnas, “On satisfying timing constraints in hard-real-time systems,” *IEEE Trans. Softw. Eng.*, vol. 19, no. 1, pp. 70–84, Jan. 1993.
[16] CiA, CAN Application Layer for Industrial Applications: CMS Service Specification, DS202-1, 1996.
[17] ——, CAN Application Layer for Industrial Applications: CMS Protocol Specification, DS202-2, 1996.
[18] ——, CAN Application Layer for Industrial Applications: NMT Service Specification, DS203-1, 1996.
[19] ——, CAN Application Layer for Industrial Applications: DBT Service Specification, DS204-1, 1996.
[20] ——, CAN Application Layer for Industrial Applications: DBT Protocol Specification, DS204-2, 1996.
[21] ——, CAN Application Layer for Industrial Applications: LMT Service Specification, DS205-1, 1996.
[22] ISO, Estelle: A Formal Description Technique Based on an Extended State Transition Model, 9074, 1997.
[23] S. Budkowski and P. Dembinski, “An introduction to Estelle: a specification language for distributed systems,” *Comput. Networks ISDN Syst. J.*, vol. 14, no. 1.
[24] S. Budkowski, Ed., *Estelle Tutorial, Amendment 1 to ISO Standard 9074*, 1989, pp. D.1–D.55.
[25] P. Dembinski, Estelle Semantics, Jun. 1986.
[26] Evry Dept., EDT Estelle development toolset, version 4.3: Estelle simulator/debugger Edb, in User Reference Manual, Inst. Nat. Télécommun. Software—Networks Dept. EVRY, France.
[27] ——, EDT Estelle development toolset, version 4.3: Estelle to C compiler EC, in User Reference Manual, Inst. Nat. Télécommun. Software-Networks Dept. Evry, France.
[28] ——, EDT Estelle development toolset, version 4.3: EDT utilities, in User Reference Manual, Inst. Nat. Télécommun. Software—Networks Dept. Evry, France.
[29] ——, EDT Estelle development toolset, version 4.3: XWindow interface for EDT XEDT, in User Reference Manual, Inst. Nat. Télécommun. Software—Networks Dept. Evry, France.
[30] ——, The Estelle/GR editor, version 2.2.1, in User Guide, Justin Templemore—Finlayson Inst. Nat. Télécommun. Evry, France.

Salvatore Cavalieri was born in Catania, Italy, in 1965. He received the Laurea degree in electronic engineering in 1989, the Ph.D. degree in electronic and computer science engineering in 1993, and a post-Ph.D. degree in electric engineering in 1995, all from the University of Catania.

He is currently a Full Professor in the Faculty of Engineering, University of Catania. His research areas include neural networks, distributed systems, real-time scheduling, and process control oriented communication protocols. He served as a member of the IEC SC65C WG6 Fieldbus standardization committee in the area of process control.