

# FLEXIBILITY, TIMELINESS AND EFFICIENCY IN FIELDBUS SYSTEMS: THE DISCO PROJECT

L. Almeida<sup>1</sup>, J. A. Fonseca<sup>1</sup>, A. Mota<sup>1</sup>, P. Fonseca<sup>1</sup>, E. Martins<sup>1</sup>,  
P. Pedreiras<sup>1</sup>, J. Ferreira<sup>2</sup>, F. Coutinho<sup>3</sup>

<sup>1</sup>DET - IEETA, Universidade de Aveiro, 3810-193 Aveiro, Portugal  
{lda, jaf, alex, pf, evm}@det.ua.pt, pedreiras@alunos.det.ua.pt

<sup>2</sup>EST, Instituto Politécnico de Castelo Branco, 6000 Castelo Branco, Portugal  
jjf@est.ipcb.pt

<sup>3</sup>Instituto Superior de Engenharia de Coimbra, 3031-601 Coimbra, Portugal  
fermaco@mail.isec.pt

***Abstract** - The requirement for flexible operation is becoming increasingly important in many distributed computer controlled systems, in order to cope with dynamic operational changes in a resource efficient way. This requirement naturally extends to the low level of such systems where, typically, field equipment such as sensors, actuators and controllers, are interconnected by a fieldbus communication system. This paper presents the DISCO project (DIStributed embeddable systems for COntrol applications) that started in September 2000, in the University of Aveiro, aiming at exploring specification, architectural and management issues in flexible distributed computer control systems for embedded applications. The project addresses the issues of control requirements definition in a QoS perspective, flexible scheduling of both network traffic as well as nodes' computing load, and global system management techniques. The paper shows the results obtained so far as well as the status of the current work in the scope of the DISCO project.*

## 1. INTRODUCTION

### 1.1 The requirement for flexibility

The requirement for flexibility is becoming increasingly important in industrial systems motivated by the need to reduce the costs of set-up, configuration changes and maintenance [1][2]. For example, reconfigurable factories that follow the flexible manufacturing concept clearly establish this requirement so that changes in production lines and manufacturing cells, necessary to move from older products to new ones, can be done with minimal down time. Furthermore, there is also a demand for operational environments that support dynamic, flexible and adaptive behaviours under timing constraints [1]. For example, it is desirable that new pieces of equipment can be installed and configured at run-time, or that the quality of system services can be negotiated on-line in order to adapt to new demands, or even that the system services can be adapted on-line to cope with evolving requirements.

This requirement for flexibility naturally extends to all system levels including those of cell and machine control, where fieldbus-based distributed computer control systems can be found. Particularly concerning the fieldbus system, flexibility also implies dynamic communication requirements so that periodic or aperiodic streams of messages can be added, removed or

adapted on-line. Simultaneously, most of the data exchanges supported by the fieldbus are subject to stringent timing constraints arising from control and monitoring requirements. However, flexibility and timeliness have typically been considered separately. Thomesse [2] states that most of the fieldbuses available today favour either one aspect or the other, i.e., either time-constrained services are guaranteed sacrificing flexibility or such guarantees are sacrificed in exchange for higher flexibility. The desired combination of flexibility and timeliness requires adequate choices of protocols and paradigms. Otherwise, it is just not possible or, at most, highly limited such as when a small number of pre-defined (static) operational modes are considered. Another requirement typically found in fieldbus systems is the capacity to deliver both periodic (time-triggered) as well as aperiodic (event-triggered) communication services under timing constraints [3]. The former ones are required to convey periodic updates of state data whilst the latter ones are required to convey alarms and management data. Again, existing fieldbus systems privilege either one or the other sort of services. In systems eminently time-triggered, event-triggered services are either non-existing or handled inefficiently in terms of either response time or network utilisation. On the other hand, in systems eminently event-triggered, interesting properties of time-triggered services such as

composability with respect to the temporal behaviour are normally lost [4]. Therefore, adequate choices are required, in terms of communication paradigms and protocols, to achieve the desired combination of both time and event-triggered services in an efficient, flexible and timely way.

There is, yet, another dimension for exploiting flexibility. In fact, the control requirements are normally established according to a desired level of performance but, when it is not possible to exactly meet such requirements, it does not mean, generally, that the system will become instable. Typically the threshold for stability is considerably away from the point of optimal performance. For many situations it might be possible to define a range of operating points in which the control performance is acceptable although below the desired optimal [5]. This leads to the definition of flexible control requirements that can be understood as QoS (Quality-of-Service) parameters. For example in a communication system, these parameters can be used for admission control negotiation to decide on whether a new stream of messages can be accepted. It might happen that in a particular situation there is not enough available bandwidth for a new stream at full rate, i.e. rate specified for highest QoS, but reducing the QoS of current streams within their ranges can release enough bandwidth for an acceptable QoS of the new stream. The concept of QoS negotiation is typical in telecommunication systems and it has been ported to several other application fields. For example, the elastic task model [6] seems well adapted to support a similar QoS negotiation to accept tasks in one processor, combining flexibility and timeliness. However, when a distributed embedded system is used, the same level of flexibility under guaranteed timeliness must be delivered by the communication system.

### 1.2 Example scenarios of *flexible* operation

Two imaginary scenarios are presented below that, despite their simplicity, allow to better visualise the requirement for flexibility highlighted above:

a) A transportation system in a factory cell has 3 conveyor belts that work in parallel. The rate at which the parts to be carried arrive at the belts is variable but often a conveyor has no part to be carried. During the time a conveyor is actually carrying a part, the speed must be adequately controlled. However, when there is no part to be carried the conveyor stops. The speed and part sensors and motor drives are interconnected to a controller by means of a fieldbus. In order to save bandwidth requirements, when a conveyor stops, the message streams carrying data from the respective speed sensors and to the respective motor drive are suspended. The same message streams are reactivated as soon as a part is detected in the conveyor input and the conveyor starts rolling. Four situations are possible concerning the dynamic status of the system: either none, one, two or three conveyors are rolling. In order to save bandwidth,

the rates of the message streams are reduced according to the number of streams currently active. For example, when 2 conveyors are rolling and the third one starts rolling too, the rates of the streams corresponding to the other two are reduced to free enough bandwidth for the new streams. However, the reduction of such rates has a negative impact on the speed control performance, i.e. QoS of the control algorithm. Thus, it has to be carried out within bounds that do not endanger the control stability.

b) A given autonomous robot has several guiding systems (e.g. line-tracking, beacon-following, wall-following, target-tracking). At run-time, the robot uses the guiding system which sensors receive the clearest input signal. This means that all systems must remain active, although only one is actually being used at a given instant in time. In this case, the sampling rate of the sensors belonging to the guiding system that is being used can be increased while the sampling rate of the remaining sensors can be decreased. Furthermore, the guiding system presently in use can increase the sampling rate of its sensors whenever difficult situations, from a control point-of-view, are detected, e.g. the line-tracking module can increase the sampling rate of the line-sensors whenever a sharp curve is detected. If the robot control system is distributed, the changes in the sensors sampling rates have to be reflected in the communication system in order to make an efficient use of the available bandwidth. This means that the communication system must cope with dynamic communication requirements.

### 1.3 About this paper

This paper deals with the issue of flexibility in distributed embedded control systems, particularly those based on fieldbuses, for example used in machine-tools with numeric control (CNCs), in industrial robotic arms, or in vehicles, either cars, trucks or automatically guided (AGVs) or more generally autonomous mobile robots. The focus is the *flexibility* at the fieldbus level, understood as the possibility to change the configuration and/or operational parameters of the communication system on-line. In general, along this paper the expression *operational flexibility* will be used mainly as a synonym for *dynamic expression of communication requirements*.

The paper presents a brief overview of previous work developed recently by the authors with the purpose of supporting flexibility and timeliness in fieldbus-based distributed embedded control systems. This work is the basis for the remainder of the paper. Then, the DISCO project (DIStributed embeddable systems for COntrol applications) is presented as well as its current status. Particularly, the paper shows the results obtained so far along the three main lines covered by the project, namely control related issues, infrastructure issues and global system management.

## 2. PREVIOUS WORK

Recently, the authors have been committed to develop ways of combining flexibility and timeliness in existing fieldbus communication systems and in bandwidth-efficient ways. From this effort, two main results were achieved: the planning scheduler, and the FTT-CAN protocol. The DISCO project, which will be described in the next section, builds upon both results and thus, a clear explanation of each is required.

### 2.1 The planning scheduler

The planning scheduler [7] is a technique to introduce operational flexibility in fieldbuses relying on static table-based scheduling. It uses a dynamic table-based concept. This scheduler scans a requirements table at run-time and builds a schedule table for a fixed period of time called a plan (fig. 1). The duration of the plan, which contains an integer number of elementary cycles, has no relationship with the periods of the message streams and thus it is not cyclic. Therefore, the plan schedule has to be rebuilt every plan. While the scheduler builds the next plan, a dispatcher scans the schedule of the current plan and triggers the transmission of the specified messages.

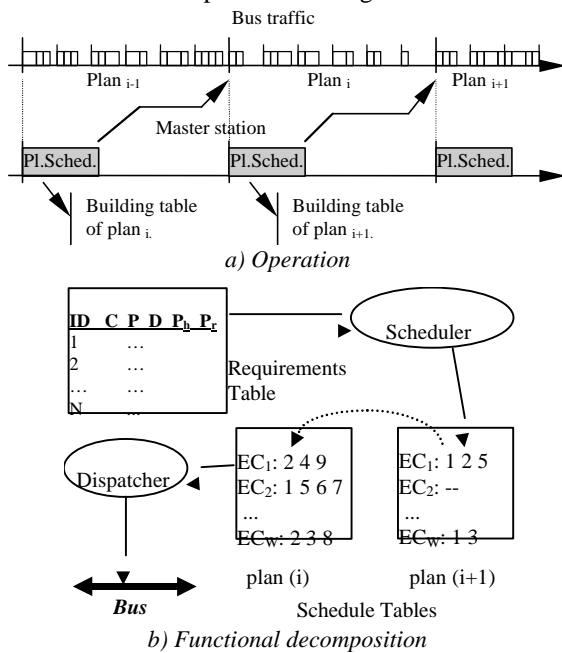


Fig. 1. The planning scheduler.

One of the main features of this scheduler is that it supports operational flexibility since on-line changes to the communication requirements are taken into account by the scheduler when building the next plan. On the other hand, it uses a scheduling look-ahead feature that allows, for example, to detect missed deadlines in advance so that corrective actions can be taken with anticipation. Another characteristic of the planning scheduler is that it is centralised and thus, the communication requirements are localised in a single node. Although that might seem negative from a fault-

tolerance perspective, it facilitates the dynamic management of the requirements table. The fault-tolerance aspect can be improved by using redundant shadow nodes running the planning scheduler and capable of taking over as soon as a failure of the active scheduler is detected. Furthermore, when fixed priorities are used, the planning scheduler imposes a lower run-time overhead on the host node than traditional on-line scheduling. This feature is highly attractive when the planning scheduler is executed on low processing power micro-controllers as those typically found in many distributed embedded systems.

### 2.2 The FTT-CAN protocol

The FTT-CAN protocol (Flexible Time-Triggered communication on CAN) aims at supporting time-triggered communication in a flexible way and with guaranteed timeliness [8]. Its flexibility is further enhanced by the capacity of the protocol to efficiently combine time-triggered with event-triggered traffic, named synchronous and asynchronous, respectively.

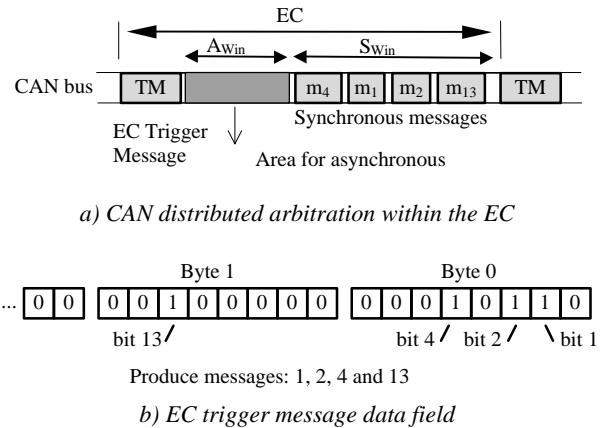


Fig. 2. Elementary Cycle (EC) structure in FTT-CAN

Basically, in FTT-CAN the bus time is slotted in consecutive fixed duration time windows called Elementary Cycles (ECs). Within each EC there are two phases (windows), a synchronous one ( $S_{win}$ ) and an asynchronous one ( $A_{win}$ ), within which the respective traffic is conveyed (fig. 2-a). Each EC is triggered by a broadcast message called EC Trigger Message, sent by a particular node called Master to synchronise all other nodes in the network. One of the distinctive features of FTT-CAN is that the scheduling for the synchronous traffic is carried out centrally on the master node. The schedule information for each EC is conveyed within the data field of the EC Trigger Message (fig. 2-b). The remaining nodes decode the EC Trigger Message and check whether they have to send a synchronous message in that EC. Two properties arise from this technique. Firstly, the centralised scheduling is achieved with very low communication overhead, just one extra message per EC. Secondly, the type of scheduling that can be executed on the synchronous traffic can now be any, independently of the prioritised native MAC protocol of

CAN, e.g. RM, DM, EDF, LLF, etc., and with relative off-sets among streams.

In order to schedule the synchronous traffic, the Master node maintains a table with the properties of the synchronous message streams called Synchronous Requirements Table (SRT). Whenever this table is changed on-line, the scheduler uses the updated information to build the schedules for the ECs. Thus, the protocol is flexible in two ways, it supports on-line changes to the communication requirements and it also allows any type of scheduling to be performed on the synchronous traffic. Furthermore, such on-line changes are subject to an on-line admission control that guarantees the continued timely behaviour of the system. Hence, the scheduling of the synchronous traffic follows a dynamic planning-based paradigm. Also, the communication control of this traffic is autonomous since transmission of messages is triggered by the communication system and not by the application. This subset of the protocol is called the Synchronous Messaging System (SMS).

Furthermore, the protocol supports event-triggered traffic (asynchronous) with autonomous control and according to a best effort scheduling paradigm. This traffic is handled in a distributed fashion such as in plain CAN. A time-guarding scheme in each node assures that this traffic is always confined to the asynchronous windows of each EC in order to enforce a strict temporal isolation between the two types of traffic. The subset of the protocol that handles asynchronous messages is called the Asynchronous Messaging System (AMS).

The asynchronous traffic can also convey real-time information, e.g. alarms, since worst-case response times can be upper bounded [9]. The capacity of the protocol to timely handle asynchronous messages is directly related to the duration of the asynchronous phase. This phase works like a periodic server, which is a known technique to handle sporadic requests in periodic systems [10]. In order to improve the real-time performance of the AMS, a minimum bandwidth can be permanently assigned to it. This is accomplished by defining a parameter that establishes an upper bound to the duration of the synchronous phases. The remaining part of each EC is then allocated to the respective asynchronous phase.

### 3. THE DISCO PROJECT

The DISCO project (DIStributed embeddable systems for COntrol applications) has been set up in order to further develop techniques, architectures and protocols for building and operating flexible distributed real-time systems that can be used in embedded control applications. A key issue is to allow exploiting the notion of control QoS to improve efficiency of resource utilisation in distributed embedded control systems.

The project started in September 2000 and has 3 years duration. It is funded by the Portuguese Government through FCT – Fundação para a Ciência e Tecnologia. It

joins several national institutions, namely IEETA - University of Aveiro, IDMEC – Instituto Superior Técnico, Lisbon, EST - Polytechnic Institute of Castelo Branco, and ISEC – Polytechnic Institute of Coimbra. The project also counts with the support of several international consultants from Lund Institute of Technology, Sweden, Institut National Polytechnique de Lorraine and Université Paul Sabatier, France and Università di Pavia, Italy.

The scope of the project is relatively wide. It considers several aspects such as the definition of QoS timing parameters from control specifications, problem partition and allocation of tasks, synchronisation techniques, minimisation of network-induced jitter, flexible real-time communication protocols, hardware scheduling co-processors, and global system management techniques. All these aspects are grouped in three complementary lines of work, respectively dedicated to control-related issues, infrastructure issues and global system management.

### 4. CURRENT PROJECT STATUS

This section presents the results that have been obtained so far, roughly one year after the project start. These results are organised according to the three main lines of work considered in the paper.

#### 4.1 Control related issues

This line of work encompasses several different aspects. In one hand, it aims at developing techniques to define QoS timing parameters based on the control specifications. This means that the period, delay and jitter requirements of the control flows will be defined with a range of possible values, instead of a fixed one. With such a feature, using an adequate architecture and protocols, it is possible to alleviate the utilization of system resources, e.g. network bandwidth or CPUs time, by reducing/adjusting the QoS of the control flows in the system. The result is an improved efficiency in resource usage. Nevertheless, no results have been obtained in what concerns this aspect, yet.

On the other hand, this line also includes the study of the impact of network-induced delays and jitter in the overall control performance and in system identification. Particularly in what concerns the latter, the aim is to check whether it is possible to take into account network-induced jitter when identifying the system so that the control parameters are tuned accordingly, leading to an improved control performance. This aspect has already known several developments. A set of experiments was carried out considering system identification techniques (e.g. least-squares, NN-based, neuro-fuzzy) and discrete control algorithms (e.g. PID, pole-placement, NN-based, fuzzy, neuro-fuzzy) distributed over a Controller Area Network [11]. The problem was studied using different priorities for the messages carrying sensor/actuator data. A jitter measurement was obtained from a set of experiments [12] in which

message delays were recorded under a traffic load based on the PSA benchmark, described in [13], showing that, under very different operational conditions, message delays in CAN follow a Gamma distribution.

The effect of jitter can be viewed as a perturbation that introduces a variable delay in the reading of the samples and in the actuation signal sent to the plant. The existence of one or both of the situations depends on the distributed control system architecture (read-in and read-out jitter [14]). The experiments in [11] show that, when jitter is not taken into account, the system identification is poor and that it can be substantially improved when jitter is taken into account, modelled as a fractional dead time. Figure 1 shows the result of simulations using the identified models for four different typical plants. The figures indicate the evolution of the sum of the squared control error using the model that does not consider jitter, upper line, and the model that accounts for jitter, lower line. Improvements between 2.3 times and 7700 times have been achieved depending on the particular plant. This somewhat surprising result seems to mean that it is possible to compensate for the effect of network-induced jitter in distributed computer controlled systems by using a more complex system model that accounts for jitter. Notice that the compensation was performed with high transmission loads and several priority arrangements between the involved messages and the remaining load.

## 4.2 Infrastructure issues

The work in this line builds upon the work described in section 2, namely the planning scheduler and the FTT-CAN protocol. The main aim is to improve the protocol towards even higher flexibility, to develop adequate message schedulability analysis and also to find new scheduling techniques to improve the network temporal behaviour in terms of both delay and jitter.

### Responsiveness of FTT-CAN to dynamic communication requirements

In what concerns the first aspect, work has been carried out to improve the responsiveness of FTT-CAN when using the planning scheduler. In the current experimental implementation of this protocol, the planning scheduler has been used within the master node to generate the EC schedules. This allows benefiting from the reduced overhead of the planning scheduler since low-processing power micro-controllers are used (the 80C592 from Philips). In this case, 8.9ms have been used for the EC duration. The plan has been set to 20 ECs resulting in duration of 178ms.

One of the limitations of the planning scheduler is that the changes performed in the Synchronous Requirements Table (SRT) take between 1 and 2 plans to actually be reflected on the bus traffic (Fig. 1). The problem is that this latency is excessive for applications where those changes have to be accomplished in just a

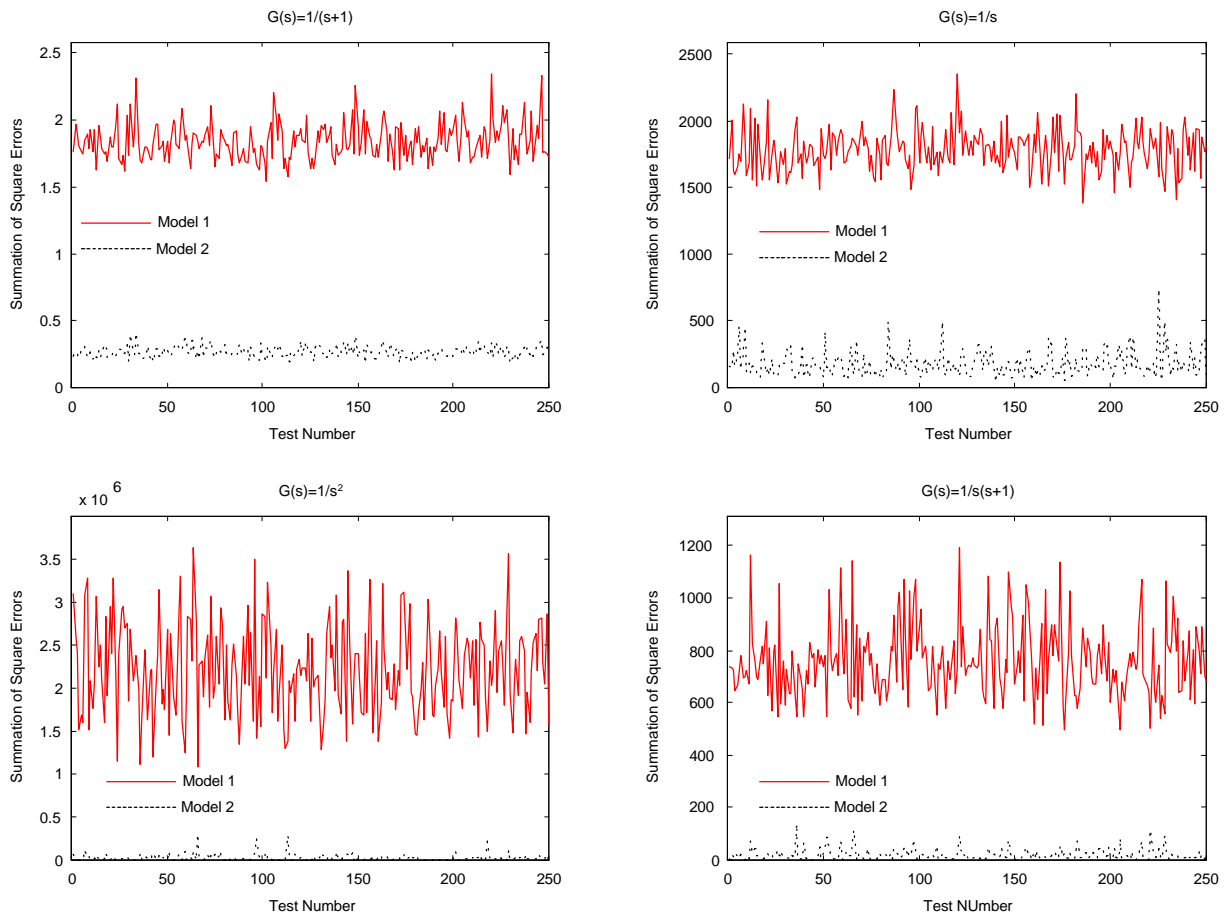


Fig. 3. Results concerning the compensation for the impact of network-induced jitter

few tens of milliseconds. The trivial solution of reducing the plan duration is not desirable because that would lead to losing the reduction in run-time overhead. Thus, a new add-on was proposed for the planning scheduler-based FTT-CAN implementation [15]. The basic idea is to use the Asynchronous Messaging System to convey the message instances that occur from the time of the change request to the time in which the change is reflected in the traffic by the planning scheduler. The new technique guarantees that a change request concerning a message with period  $P$  will be reflected in the bus traffic at most after  $P+1$  (in ECs), independently of the plan duration.

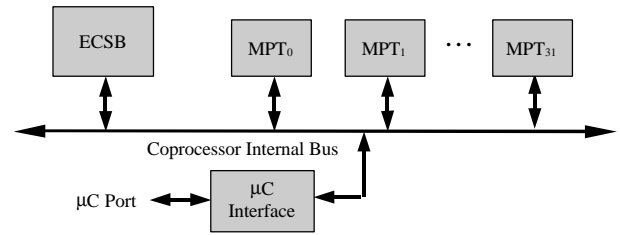
#### H/W scheduling support for the FTT-CAN Master

Apart from the limitations in terms of responsiveness, the planning scheduler still presents a considerable overhead for simple 8-bit microcontrollers. In this case, although it has a reduced overhead when compared to fully dynamic approaches, the fact is that the master node has practically no time for any other function but the execution of the planning scheduler. This is an impediment to run other functions such as error monitoring and admission control in the master node. Therefore, to relief the master node microcontroller from the burden of executing the planning scheduler, a preliminary specialised FPGA-based scheduling coprocessor named PSCoP was designed [16]. In this case, the scheduling function is performed on the fly in much less than one EC. An initial version of this coprocessor based on an XC4010 FPGA from Xilinx, clocked at 12MHz, was capable of scheduling a set of 8 periodic message streams for a plan with a length of 16 ECs with a worst-case execution time of  $63\mu\text{s}$ .

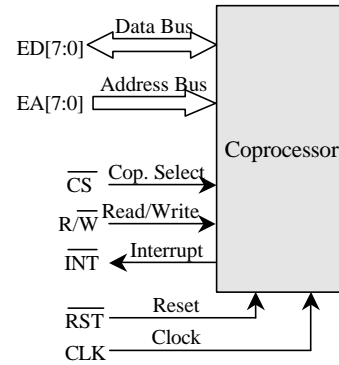
Building upon the preliminary experience gained with the PSCoP design, another coprocessor has just been designed that operates dynamically instead of following the planning scheduler approach [17]. This allows for an increased responsiveness since any changes in the SRT can now be reflected on the bus in the following EC. This version has increased features such as the possibility to choose the scheduling policy between rate monotonic, deadline monotonic or arbitrary priorities, and the inclusion of a schedulability analyser based on the timeline method [18]. This method allows checking the schedulability of a message set just by using the basic scheduling unit. Also, the number of messages in the set has been increased to 32 and the interface with the node CPU has also been improved. Figure 4 shows its internal architecture (a) and external interface (b).

In terms of performance, for a scenario based on a data rate of 1Mbit/s, an EC duration of 1ms and a coprocessor clock frequency of 20MHz, the new coprocessor is expected to build an EC schedule in less than  $14.6\mu\text{s}$ , or about 1.5% of the EC time. The time taken to assess the schedulability of a message set is highly dependent on the properties of the message streams. However, experiments carried out with the SAE benchmark, described in [25], showed that the

coprocessor takes at most 1/3 of the EC (5ms in this case) to check the schedulability of this message set.



a) Coprocessor architecture (ECSB- EC Schedule Builder; MPT - Message Production Timer)



b) Coprocessor external interface.

**Fig. 4.** Enhanced FPGA-based scheduling coprocessor for FTT-CAN.

#### Analysis of traffic temporal behaviour

Concerning the analysis of the traffic temporal behaviour, in particular synchronous traffic, work has been carried out to accommodate the particular features of the underlying scheduling model of FTT-CAN. Two distinctive features were taken into account, non-preemption of message transmission and insertion of idle-time to enforce the regularity of the ECs. The resulting schedulability analysis has been generalised and presented in [18]. As for the asynchronous traffic, recent work [19] has enlarged the analysis presented in [9], creating three categories, A1, A2 and A3:

- A1:** hard real-time sporadic messages with deadlines less or equal to the respective minimum inter-arrival time;
- A2:** hard-real time sporadic messages with deadlines greater than the period, which require buffering;
- A3:** soft and non-real-time sporadic messages.

Particularly, the work in [19] not only presents an improved analysis for class A1, with respect to the one presented in [9], but it also expands the analysis for class A2, which allows calculating the maximum number of buffers that are required in the sender nodes as well as the worst-case response time to a transmission request.

#### Improving the traffic temporal behaviour

In what concerns new scheduling techniques to improve

the temporal behaviour of the network, two directions have already been pursued. The first one, started recently, investigates ways to deal with network errors in FTT-CAN in order to minimise their impact [20]. This work considers errors during the transmission of synchronous and asynchronous messages, only. Errors in the transmission of EC trigger messages are not yet considered. The approach proposed in the paper is based on the enforcement of the duration of each phase within the EC so that any transmission errors do not cause delays beyond the EC in which they occurred. This results in an error confinement to the EC boundaries. Furthermore, the paper proposes an interaction between the error detection unit and the traffic scheduler so that the erroneous transmission can either be rescheduled and retransmitted later still within its deadline, or rejected, or even trigger a recovery procedure (fig. 5).

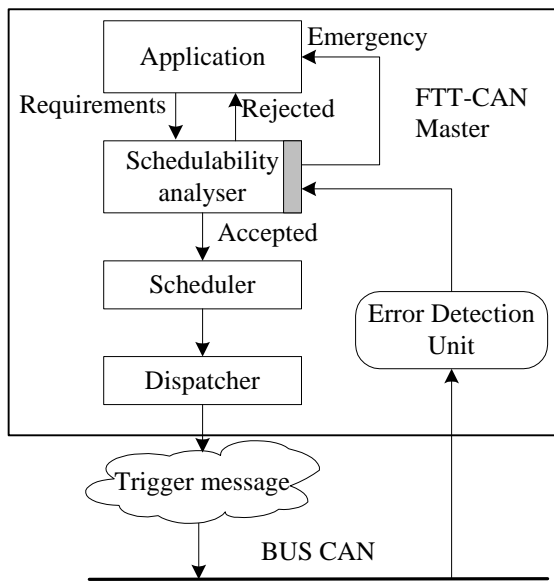


Fig. 5. Timely error handling in FTT-CAN.

On the other hand, work has been carried out towards minimisation of network-induced jitter using genetic algorithms [21]. Basically, it determines adequate initial offsets to apply to the message streams, setting them out-of-phase and reducing, or even eliminating, the mutual interference caused by scheduling over a shared medium. For example, this technique has been successfully applied to the SAE benchmark, delivering the right offsets for a jitter-free (or minimized) operation. It is also noteworthy the fact that a new technique was developed to execute the genetic algorithm in a progressive fashion (proGA), resulting in a considerable reduction in computational demand and in the time required to compute a solution. This technique consists on minimizing the jitter for each message stream in a priority decreasing order, keeping the initial offsets already computed, i.e. of the higher priority messages, and determining only one new offset at a time. The negative side of this technique is a lower efficiency in the search for an optimal solution. However, since higher priority message streams are

those that cause the highest interference, minimizing the jitter of these streams, which is accomplished fast with the proGA technique, leads to near optimal solutions. A recent work [23] discusses the application of this technique to TT-CAN, a Time-Triggered profile for CAN that is in the process of being standardised by ISO [24].

### 4.3 Global system management

In this line of work, the main purpose is to develop efficient ways of managing the flexibility of the distributed infrastructure. This means developing services to support dynamic requirements, namely communication ones, to interface with an operator, to execute the admission control and to synchronise related activities, either tasks and messages. In this direction, some work has already been carried out concerning the FTT-CAN protocol, which has been presented in [22] and named network-centric approach to global system management.

#### A detachable operator console

An important concept introduced in that work is the Operator Console. This is a special node that executes the operator interface, delivering services to manage the communication system such as change, add and/or delete message streams. Moreover, this node also executes the admission control whenever the changes to the communication requirements imply an increase in bandwidth utilisation (fig. 6).

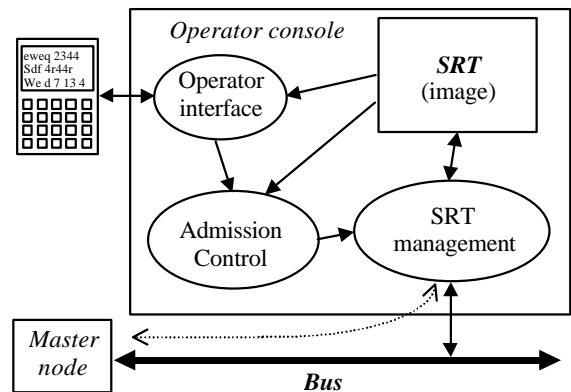


Fig. 6. The operator console.

The Operator Console (OC) maintains a copy of the Synchronous Requirements Table (SRT). An appropriate confirmation protocol allows to assure the coherence of this copy with respect to the original SRT in the Master node. The OC is only required for managing the system and thus, can be disconnected when it is not necessary. This aspect seems particularly adequate to systems that are meant to be embedded in machines, vehicles or robots, reducing the number of functions permanently residing in the system and consequently the system cost. However, this approach can be followed when the requests for changes on the streams are issued by an operator, only. In the case of more dynamic systems that are expected to react to

change requests imposed by the environment, the admission control must reside permanently in the system, and in the Master node for efficiency reasons.

### The network-centric approach

The network-centric approach considers the fieldbus within a distributed computer control system as an independent fundamental element. Its distinguishing property is that it interfaces with all the nodes and thus, it is particularly well positioned to support the system global control. In fact, the fieldbus appears as the only common infrastructure when considering all the distributed activities being executed over the system.

In a distributed system using autonomous time-triggered communication, a given producer node  $k$  executes a task  $A$  that continuously samples some state variables of the environment and produces the respective values (i.e. makes them available in the network interface). The communication system, in turn, cyclically transmits a message  $M$  to broadcast those values. In the consumer node(s)  $n$  a task  $B$  continuously consumes those values and uses them to generate some output. Notice that there is no direct relationship between the rates at which the three activities,  $A$ ,  $B$  and the transmission of  $M$  are executed. Therefore, the flow of information from node  $k$  to node  $n$  progresses at a rate determined by the slowest of the three.

An efficient approach in what concerns CPU and network utilisation is to force the three rates to be equal using some synchronisation mechanisms. For example, this can be achieved by forcing the production / consumption services to wait for the respective message transmission / reception. In this case, all the distributed cyclic activities in the system execute at a rate imposed by the transmission of messages (fig. 7). Hence, by controlling the communication requirements specified in the communication system, an operator can easily control the rates of the different information flows and associated activities being processed in the system. Furthermore, this level of control can also be applied to the relative phasing of those flows, a feature that is not possible with event-triggered communication, due to the asynchronous nature of events. Finally, due to the synchronism between tasks execution and messages transmission, the network-centric approach seems particularly adapted to support an holistic admission control that takes into account the availability of system resources in an integrated fashion, both network and CPUs bandwidth.

## 5. CONCLUSION

This paper discusses the requirement for flexibility felt in industrial distributed computing systems and, particularly, those based on fieldbus communication systems. Previous work that has been developed at the University of Aveiro to combine traffic timeliness and flexibility in a resource efficient way is briefly described, namely the planning scheduler and the FTT-CAN protocol. The paper presents, then, the DISCO project (DIStributed embeddable systems for Control applications) that is being developed in that institution aiming at exploring specification, architecture and management issues in flexible distributed computer control systems for embedded applications. The project addresses the issues of control requirements definition in a QoS perspective, flexible scheduling of both network traffic as well as nodes' computing load, and global system management techniques. The paper integrates and shows the results obtained so far in the scope of the DISCO project.

In what concerns the control related issues, a preliminary study has been carried out quantifying the improvement in plant control performance when network-induced jitter is accounted for, in the system identification, as a fractional dead-time. The infrastructure issues have known a considerable development around FTT-CAN. An appropriate protocol has been developed to enhance the system responsiveness to change requests in the synchronous requirements when the planning scheduler is used. Also, an FPGA-based coprocessor has been designed to boost system responsiveness and minimize the run-time overhead of dynamic scheduling approaches. This coprocessor has several interesting features notably the merging of the scheduler function with a schedulability analyser. An adequate analysis of the protocol temporal behaviour has been carried out that lead, for example, to the proposal of the timeline technique that is particularly efficient for use in the scheduling coprocessor referred before. A preliminary study on the handling of transmission errors in FTT-CAN has been developed, too. Also, the use of genetic algorithms has been proposed to generate adequate offsets for the message streams in order to minimize scheduler-induced jitter. Finally, an integrated approach has been proposed for the dynamic management of tasks and messages in FTT-CAN, centred on the network and named network-centric approach. The underlying idea is to use

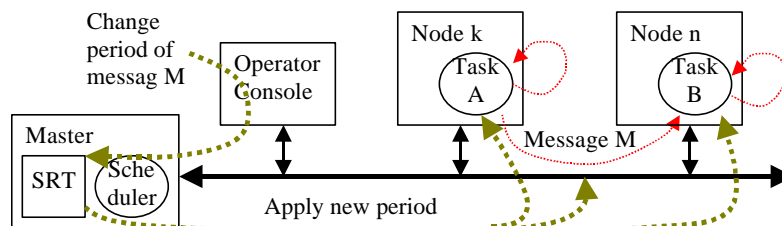


Fig. 7. Synchronising tasks activations with network traffic

communication services, in the tasks, which are synchronised with the traffic that is autonomously managed by the communication system. This approach will be used to support a holistic analysis of the system, so that dynamic requirements are considered based on the availability of either computing as well as communication resources.

## REFERENCES

- [1] J. Stankovic *et al.*. Strategic Directions in Real-Time and Embedded Systems. *ACM Computing Surveys*, **28(4)**: 751-763, 1996.
- [2] J.P. Thomesse. The Fieldbuses. *Annual Reviews in Control*, **22**: 35-45, 1998.
- [3] J.P. Thomesse. Main Paradigms as a Basis for Current Fieldbus Concepts. *Proc. FeT '99 - Fieldbus Systems and Applications Conference*, Magdeburg, Germany, September 1999.
- [4] H. Kopetz. *Real-Time Systems Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.
- [5] K. Shin, *et al.* Adaptation and Graceful Degradation of Control System Performance by Task Reallocation and Period Adjustment, *Euromicro Conf. Real Time Systems*. York, UK. June 1999.
- [6] G. Buttazzo, G. Lipari and L. Abeni. Elastic Task Model for Adaptive Rate Control. *Proc. of RTSS'98 (19th IEEE Real-Time Systems Symposium)*, Madrid, Spain. December 1998.
- [7] L. Almeida, R. Pasadas and J.A. Fonseca. Using a planning scheduler to improve the flexibility in real-time fieldbus networks. *IFAC Control Engineering Practice*, **7**: 101-108, February 1999.
- [8] L. Almeida, J.A. Fonseca and P. Fonseca. A Flexible Time-Triggered Communication System Based on the Controller Area Network: Experimental Results. *Proc. FeT'99 - Fieldbus Systems and Applications Conference*, Magdeburg, Germany. September 1999.
- [9] P. Pedreiras and L. Almeida. Combining Event-Triggered and Time-Triggered Traffic in FTT-CAN: Analysis of the Asynchronous Messaging System, *WFCS'2000 (IEEE Work. on Factory Communication Systems)*, Porto, Portugal, 5-8 Sept 2000.
- [10] B. Sprunt, L. Sha and J. Lehoczky. Aperiodic Task Scheduling for Hard-Real-Time Systems. *Journal of Real-Time Systems*, **1**, 1989.
- [11] A. Mota and J. A. Fonseca. Systems Modelling and Identification in CAN based Distributed Control Systems. *DCCS 2000: 16th IFAC Workshop on Distributed Computer Control Systems*. Sydney, Australia, 29 November-1 December 2000.
- [12] L. Almeida, P. Fonseca, J.A. Fonseca and Z. Mammeri. Scheduling and Clock Synchronisation in CAN-based Distributed Systems. *CiA ICC'99 (Int. Conf. on CAN)*, Turin, Italy. November, 1999.
- [13] N. Navet and Y.-Q. Song. Performance and Fault Tolerance of Real-Time Applications Distributed over CAN (Controller Area Network), *CiA - CAN in Automation Research Award*, 1997.
- [14] Stothert, *et al.* Effect of Timing Jitter on Distributed Computer Control System Performance. *Proc. DCCS'98 (15<sup>th</sup> IFAC Workshop Distrib. Comp. Control Systems)*, Sept. 1998.
- [15] P. Pedreiras, L. Almeida and J. A. Fonseca. Improving the responsiveness of the synchronous messaging system in FTT-CAN. *DCCS 2000 (16th IFAC Work. on Distributed Computer Control Systems)*. Sydney, Australia, 29 Nov – 1 Dec, 2000.
- [16] E. Martins, P. Neves and J. Fonseca. PSCoP – A Planning Scheduler Coprocessor. *WIP Session of WFCS'2000 – 3rd IEEE Int. Work. on Factory Communication Systems*. Porto, Portugal, 5-8 Sept 2000.
- [17] E. Martins and J. Fonseca. Improving Flexibility and Responsiveness in FTT-CAN with a Scheduling Co-processor. *Proc. FeT'01 - Fieldbus Systems and Applications Conference*, Nancy, France. Nov 2001.
- [18] L. Almeida and J.A. Fonseca. Analysis of a Simple Model for Non-Preemptive Blocking-Free Scheduling. *ECRTS'01 (Euromicro Conf. on Real-Time Systems)*, Delft, Holland. June, 2001.
- [19] P. Pedreiras and L. Almeida. Asynchronous Communication on FTT-CAN: Experimental Results. *Proc. FeT'01 - Fieldbus Systems and Applications Conference*, Nancy, France. Nov 2001.
- [20] J. Ferreira, P. Pedreiras, L. Almeida and J.A. Fonseca. FTT-CAN Error Confinement. *Proc. FeT'01 - Fieldbus Systems and Applications Conf.*, Nancy, France. November 2001.
- [21] F. Coutinho, J.A. Fonseca, J. Barreiros, E. Costa - Jitter Minimisation with Genetic Algorithms. *WFCS'2000, IEEE Workshop on Factory Communication Systems*, Porto, Portugal, 5-8 Sept 2000.
- [22] L. Almeida and J.A. Fonseca. FTT-CAN: a Network-Centric Approach for CAN based Distributed Systems. *SICICA'00 (4th IFAC Symp. on Intelligent Components and Instruments for Control Applications)*, Buenos Aires, Argentina, 13-15 September 2000.
- [23] F. Coutinho, J.A. Fonseca and J. Barreiros. Scheduling for a TT-CAN Network with a Stochastic Optimisation Algorithm. *Proc. FeT'01 - Fieldbus Systems and Applications Conference*, Nancy, France. November 2001.
- [24] ISO/WD11898-4. *Road Vehicles – Controller Area Network (CAN) – Part 4: Time-Triggered Communication*, December 2000.
- [25] K. Tindell and A. Burns. Guaranteeing Message Latencies on Control Area Network (CAN). *CiA ICC'94 (Int. Conf. on CAN – Controller Area Network)*, Mainz, Germany, 1994.