

ISSUES ON TASK DISPATCHING AND MASTER REPLICATION IN FTT-CAN

José A. Fonseca*, J. Ferreira**, M. Calha**, P. Pedreiras*, L. Almeida*

* DET – IEETA, Universidade de Aveiro, ** EST, Instituto Politécnico de Castelo Branco, Portugal

ABSTRACT

The FTT-CAN (Flexible Time-Triggered Communication on Controller Area Network) protocol supports time-triggered communication in a flexible way as well as the combination of both time and event-triggered traffic with temporal isolation. Different previous papers have already discussed its potentialities and presented worst-case temporal analysis for both types of communication. In this paper, after a brief review of the main characteristics of the protocol, new issues concerning its use in distributed embedded systems are presented: the extension for task dispatching and the inclusion of techniques to improve fault tolerance, namely master replication.

1. INTRODUCTION

Many distributed systems either for automotive applications or embedded in simpler machines or devices rely in low processing power microcontrollers to implement the functionalities required for each node. Besides the activities related to functionalities such as data acquisition, actuation, pre-processing of raw data and other process dependent operations, these controllers are also subject to processing overhead to handle communications. It is then difficult to include a real-time kernel if multitask at the node level is desired. Thus, the possibility of controlling the dispatching of the tasks in the different nodes without imposing a significant overhead is an interesting issue.

A seminal paper published by Tindell *et al* in 1994 [1] introduced the idea of the joint scheduling of tasks and messages, naming it holistic scheduling as the overall system was treated as a whole. The perspective was to verify the end-to-end response times for distributed real-time software systems, particularly the ones conforming to a TDMA architecture.

During 2001, Richard *et al* [2] addressed again the holistic perspective. In their paper an optimal priority assignment for fixed-priority tasks and messages in an automotive computerised system is presented. The target architecture is based on multiple fieldbus networks connecting uniprocessor computation units. Tasks and messages are on-line scheduled according to fixed priorities. The priority assignment is performed with a branch and bound algorithm.

In what concerns this paper, in section 3, a very simple solution for the holistic dispatching of tasks and messages is presented. This solution is based on the particularities of the FTT-CAN protocol [3,4], which has been developed by the DES – Distributed Electronic Systems team at the research unit IEETA in the University of

Aveiro. A brief review of this protocol is included, in section 2.

Various aspects of message scheduling in FTT-CAN have been studied by the team during the last three years, including, recently, fault tolerance issues [12][13]. The new approach of using FTT-CAN for task dispatching reinforces the need for fault tolerance. Some initial work concerning one of the important aspects in this domain, the master replication, is also presented and discussed in this paper, in section 4.

In section 5 of the paper, the conclusions are presented and several on-going research works to validate the preliminary ideas here discussed are identified.

2. THE FTT-CAN PROTOCOL

Flexible operation is a rather important requirement in modern industrial systems and it has to be supported at all system levels, including the field level in process industry, and the cell and machine control levels in manufacturing industry, where fieldbus-based communication systems are commonly found. Typical applications at these levels require both time and event-triggered communication services, in most cases under stringent timing constraints, to convey state data in the former case and alarms and management data in the latter. However, neither the requirement for flexible operation under guaranteed timeliness nor for joint support of time and event-triggered traffic are efficiently fulfilled by most of existing fieldbus systems as it is discussed in [4].

FTT-CAN (Flexible Time-Triggered communication on Controller Area Network) is a recent protocol which fulfils both requirements. It supports flexible time-triggered communication and an efficient combination of both time and event-triggered traffic with temporal isolation. These types of traffic are handled by two complementary subsystems, the Synchronous and the Asynchronous Messaging Systems, respectively. In [4], a justification for the needs that led to the development of a new protocol as well as its detailed description and the worst-case temporal analysis for both subsystems are also presented, showing the capability of the protocol to convey real-time traffic of either type.

FTT-CAN supports synchronous messages transmitted periodically and autonomously by the communication system. A requirements table (Synchronous Requirements Table - SRT) describes the periods (P), deadlines (D), sizes (S), initial phasing (Ph) and priorities (Pr) of the synchronous messages:

$$SRT = \{m_i = m_i(P_i, D_i, S_i, Ph_i, Pr_i), i = 1..N_{SRT}\}$$

These messages are guaranteed to meet their deadlines by using an appropriate on-line schedulability analysis. In

what concerns asynchronous messages, they are handled by the communication system in an event-triggered fashion according to a best-effort approach. However, offline, it is possible to determine worst-case response times for these messages, allowing their use to support sporadic hard real-time data transfers. The protocol assures a clear separation between synchronous and asynchronous messages so that there is no negative interference of the latter ones on the timeliness guarantees for the former ones.

In FTT-CAN the bus time is broken down into fixed duration slots called elementary cycles (EC) composed of two phases, synchronous and asynchronous. All periods and deadlines are then expressed as integer multiples of the EC duration (defined at start-up). A special node, the master, holds the SRT and it is responsible for scheduling and dispatching. Every elementary cycle is initiated by the EC trigger message, sent by the master, which carries the identification of the synchronous transactions that must be carried out during that EC (Figure 1-a)).

The identification of each message is done by associating to it a one bit flag. The state of the flag indicates if the correspondent message should or not be transmitted in the EC. These bit flags are transmitted in the data field of the trigger message.

The involved producers respond to the call issued by the trigger message trying to transmit their messages concurrently within the synchronous phase. Contention on the bus is left for the CAN native arbitration to sort out (fig. 1-b)).

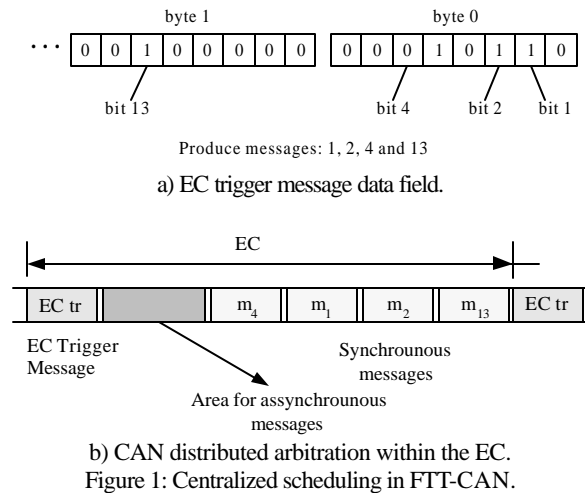


Figure 1: Centralized scheduling in FTT-CAN.

The EC duration is taken into account by a centralized scheduler, running on the master node, so that all transactions scheduled for the same EC are guaranteed to fit in. The protocol reserves a range of high priority identifiers for this message.

The FTT-CAN protocol defines two non-overlapping (isolated) windows within each EC for the transmission of synchronous and asynchronous traffic respectively (Figure 1-b). This isolation is enforced at all nodes competing to send asynchronous messages in order to guarantee that these do not interfere with the timeliness of synchronous ones.

To achieve flexibility in what concerns on-line changes to the synchronous requirements table SRT, a dynamic scheduler must be used. However, dynamic scheduling brings along a significant run-time overhead which, on low-processing power microcontrollers, would necessarily cause an impediment to an efficient network bandwidth utilization. Thus, in order to reduce the run-time overhead, the FTT-CAN protocol can be combined with a dynamic table-based scheduler known as planning scheduler [5]. This scheduler scans the requirements table SRT and produces a schedule table for a fixed duration period of time called a plan. The duration of the plan is independent of the messages' periods and so the plan is not cyclic in general, needing to be rebuilt periodically. Thus, changes to the SRT table can be accounted for when building the next plan.

Another approach using an FPGA based specialized coprocessor, the Planning Scheduler Coprocessor [6], has also been developed. In this solution the impact of dynamic scheduling is strongly reduced and the planning scheduler does not need to be used, since the coprocessor is capable of scheduling ECs on the fly.

3. TASK DISPATCHING WITH FTT-CAN

In what concerns this paper, a very simple solution for the holistic dispatching of tasks and messages is presented. This solution is obtained by slightly adapting the FTT-CAN protocol to include the trigger of task dispatching in remote nodes, besides triggering just the transmission of messages as it was explained in section 2. To do this, the FTT-CAN trigger message transmitted periodically by the master includes a set of additional flags besides the ones used for messages and the reserved bytes.

In figure 2 an example of the new distribution of the bit flags in the trigger message is shown. As in the message case, whenever a logic "1" is sent in a specific flag, the correspondent task should be dispatched in the node where it must be executed.

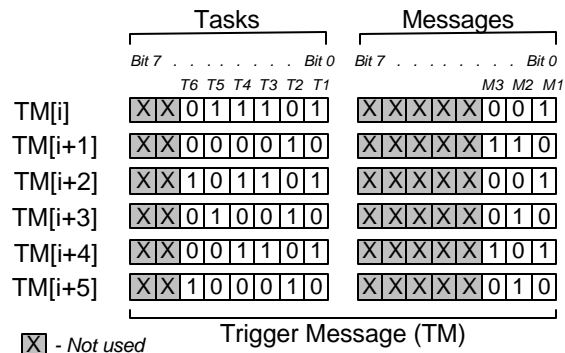


Figure 2: Example of trigger messages including task dispatching flags

As there must be one flag for each synchronous message and for each task in the system, the 64 bits maximum data length of a CAN message may become insufficient for medium or large sized systems. A solution for this consists in transmitting a second contiguous trigger message, thus

extending the number of available bits to 128. Considering the need for time resolution in real world distributed systems, which conditions the minimal EC duration, this solution is feasible in the sense that the protocol overhead for typical scenarios range from 3 to 6% (CAN at 500 Mbits/sec, EC with 10ms).

Another important issue in what concerns the limitation in the number of flags is that this dispatching technique is not specific of CAN. Recent work at Aveiro [7] has extended the flexible time-triggered approach to message dispatching in distributed systems based in other communication infrastructures, particularly Ethernet. It is obvious that in this case there is not the same limitation in the number of bits as in CAN, so, again, the inclusion of task dispatching is possible.

This approach can be generalised for almost any type of systems where multicast or broadcast of messages is possible. The price to pay to include task dispatching is just one additional bit per task, which seems to be almost insignificant even in protocols where the length of useful data in each message is low as in CAN.

An interesting feature of this solution is the simplicity of the kernel at the nodes, called a nano-kernel [8]. In fact it can just be an interrupt driven piece of code adapted from the one developed for FTT-CAN. Upon a trigger message interrupt, first the messages to produce are sent to the transmission buffers and after the control of the CPU is passed to the task whose trigger flag is set, if any. Some particular characteristics of the solution should be referred:

- The trigger message can convey more than one flag set for the same node.
- The tasks can be prioritised at the node level or at a system level.
- A task triggered in an EC can preempt a running task if its priority is higher than the priority of this one.
- Several tasks can be executed in sequence at a node if triggered together in the same EC.
- If required, a local scheduler can be included at the node, with the trigger message being used to mark the tasks' release instants (this implies additional processing overhead and may result in the need for more powerful CPUs at the nodes).

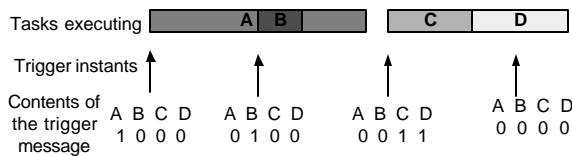


Figure 3: Example of task execution at the node level

It can be seen that the mechanisms needed at the nano-kernel to implement the previous features are rather simple and well adapted to be handled by the typical controllers used in automotive and other embedded applications.

The other advantage of this solution is that it may in the future extend the flexibility obtained for message scheduling to task scheduling. The flexibility in message scheduling comes from the use of on-line admission control

at the master node. This node can be used to receive demands for changes in the message set or in the messages' parameters and, upon the results of a verification of schedulability, accept or reject those changes. If accepted, they can be introduced in the dispatching procedure some ECs later or even almost at once, depending on the solution used at the master node. If a software planning-based solution is used then it takes some time to decide and to include the changes. However, this time can be shortened if the asynchronous message windows are used to transmit the first instances of a new message stream [9]. If a hardware-based solution is used [10], the inclusion of the new message stream in the system can be taken into consideration just one EC after the change request.

At this moment the on-going work consists mainly in adapting the master to accept a static task dispatching table. This is rather simple for both solutions (software or hardware). The problem of the joint scheduling will be soon addressed. The use of a hardware co-processor [10] to accelerate admission control and scheduling at this level is one of the ideas to explore.

4. MASTER REPLICATION in FTT-CAN

As it was seen above, FTT-CAN relies in a special node called the master for a set of critical operations: admission control, scheduling, dispatching. The extension to task dispatching just increases this criticality. Therefore, in order to prevent such situation, a backup master must be used. A backup master must monitor the network looking for EC trigger messages. Whenever the next trigger message is delayed more than a given tolerance the backup enters into action and transmits the missing EC trigger message. From that moment on, the backup must become the primary master and the previous primary, if still active, must become a backup. Notice that more than one backup master may be used, as long as each one is assigned a different priority.

The physical replication of the master nodes and the timing mechanisms to detect missing trigger messages are not enough to guarantee a correct on-line replacement of the master. The backup masters must have a coherent copy of the requirements table (SRT). They must also be able to produce exactly the same outputs (schedules which are reflected in the trigger messages) as the primary master in charge, in order to guarantee replica determinism. An important aspect is then the synchronization between primary and backup masters.

The backup master can verify autonomously that synchronization is required. In every EC all backup masters compare their own schedules with the schedule conveyed in the trigger message. Whenever an inconsistency is detected the backup master issues a synchronization request, causing the current primary master to download the SRT as well as the relative phasing information necessary to resume scheduling synchronously. This process can be done in two different ways, depending on the type of operation of the FTT-CAN based system.

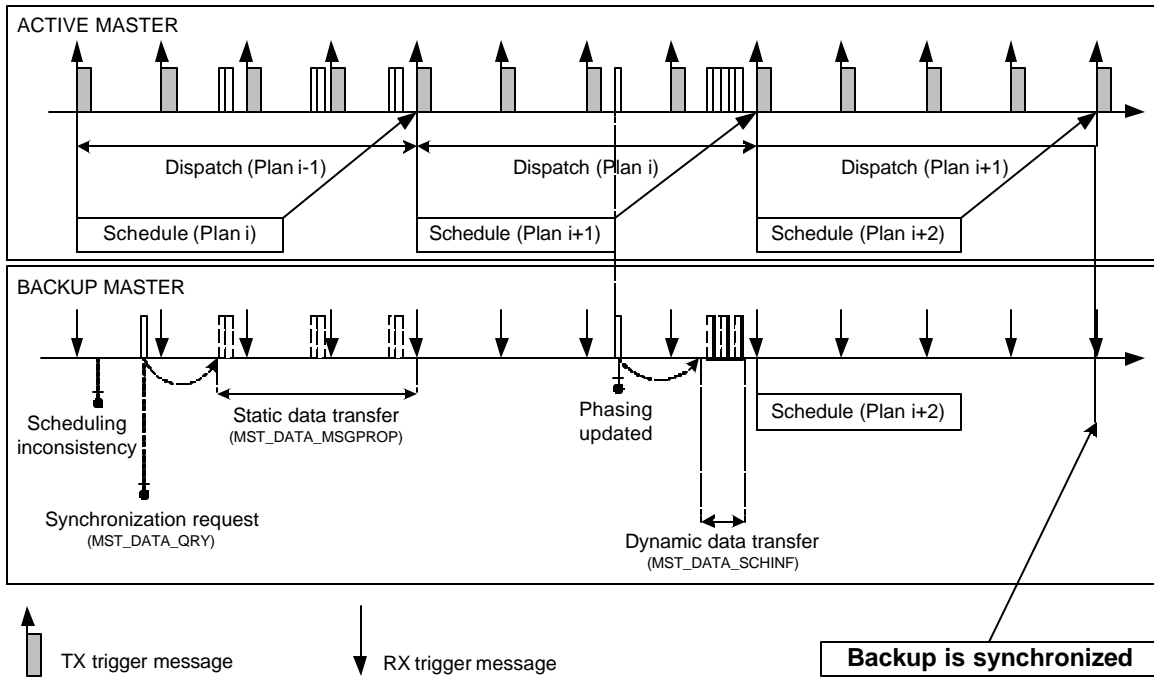


Figure 4: Timeline of the synchronization process for a planning-based system.

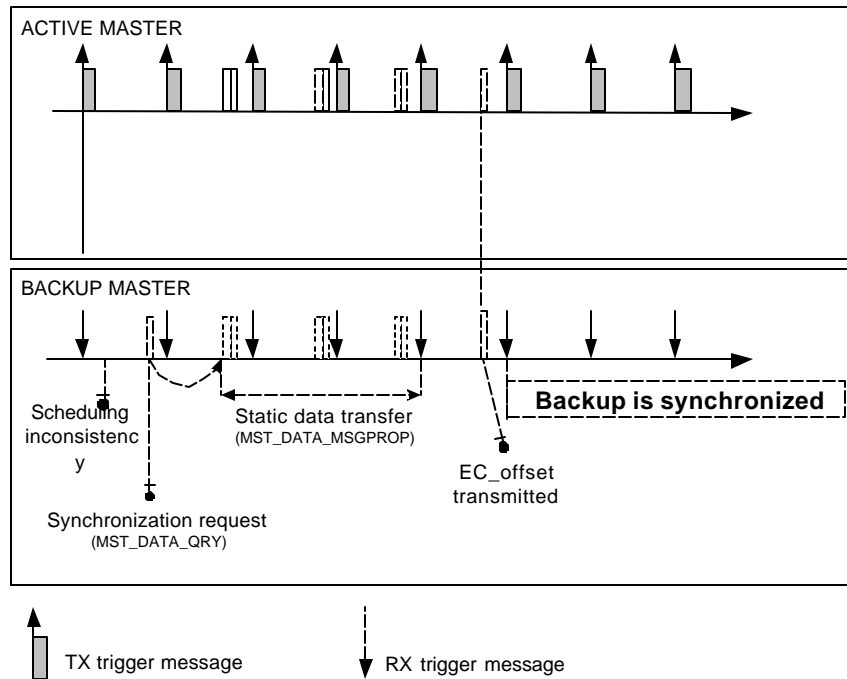


Figure 5: Synchronization process for masters with an EC by EC scheduler.

For a planning based approach [5], once the active master receives the synchronization request (MST_DATA_QRY), it starts to download the SRT table and the relative phasing data in two rounds. In the first round, the SRT is split and conveyed into several messages (MST_DATA_MSGPROP). These messages

only carry the static information (e.g. period, deadline, message IDs, etc). Once the first state transfer round is complete, the dynamic scheduling dependent data (e.g. relative phasing) is also split into several messages (MST_DATA_SCHINF). The transmission of this last state transfer round must be enclosed within a single

plan and only after the scheduling of the next plan is completed in order to assure the consistency of the time dependent scheduling data. Once the scheduling dependent data is fully received by the backup master, it waits for the beginning of the next plan to start the scheduling. After completing the scheduling of the next plan, the backup master is ready to monitor the trigger messages produced by the active master and replace it in case of failure, as soon as a new plan begins. The timeline of the overall synchronization process is depicted in figure 4.

For an EC by EC scheduling approach such as the one used when a hardware co-processor is available at the master node, a solution is foreseen in which the synchronization can take less time. It consists in adding to the co-processor an EC counter which permits to determine, in each EC, the offset (EC_OFFSET) relative to the start of the scheduling operation. This offset should be obtained from the last static information available, i.e., either from the beginning of operation or from the last accepted change in the message set (new message accepted, message eliminated, change in message parameter, ...). After the transfer of the static data, the EC_OFFSET value can be transferred at once and the co-processor can start an internal synchronization procedure during which it calculates the scheduling from the beginning till the current EC. This procedure is identical to the schedulability analysis described in [11]. It can also be done in just a fraction of the EC duration.

The synchronization process is a time critical task since, during its execution, modifications of the SRT are not allowed and the system may work without a backup master (if just one replica exists). The process of transmitting the static information, which must be done in both previous approaches, can take a few ECs depending on the size of the SRT and on the current network utilization. It is then important to develop new solutions to accelerate the synchronization, such as the one proposed above.

Another issue currently under study is the problem of a failure in the primary master during a synchronization process. To tolerate this failure there must be at least 3 redundant masters and it is mandatory that one of the backup masters keeps synchronized all the time. This can be made possible if all the changes of the SRT are done through the network. Backup masters can then monitor those operations and keep always synchronized in the absence of exceptional situations. Changes at the requirements table SRT issued locally at the primary master can not, in consequence, be tolerated.

5. CONCLUSIONS

This paper presents a new solution, based in the FTT-CAN (Flexible Time-Triggered communication on Controller Area Network) protocol, which provides joint dispatching of tasks and messages thus enabling complete synchronisation in the operation of the nodes in a distributed system. The solution is well adapted to the low-processing power microcontrollers frequently

used in typical embedded applications due to the low overhead it imposes in most of the system nodes. However, it relies on a specific node, the master, which has extended functionalities.

The problem of achieving fault tolerance, common to the use of FTT-CAN just for message scheduling, passes then, among other techniques, by replicating the master node. This paper addresses also the issue of master replication with emphasis in the synchronisation processes needed to obtain replica determinism. This is particularly important when the set of requirements can be changed during system operation as it is the case in FTT-CAN.

REFERENCES

- [1] Tindell, K., Clark, J.: "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Microprocessing & Microprogramming* 40, 117-134, 1994.
- [2] Richard, M., Richard, P., Cottet, F.: "Task and Message Priority Assignment in Automotive Systems", *4th IFAC Conference on Fieldbus Technology (FET'01)*, Nancy, France, November 2001.
- [3] Almeida, L., Fonseca, J., Fonseca, P.: "Flexible Time-Triggered Communication on a Controller Area Network", *Proc. of Work-In-Progress Session of RTSS'98 (19th IEEE Real-Time Systems Symposium)*, Madrid, Spain, December 1998.
- [4] L. Almeida, Pedreiras, P., Fonseca, J.A.: "The FTT-CAN protocol: Why and How", to appear in *IEEE Transactions on Industrial Electronics*, 2002.
- [5] Almeida, L., Pasadas, R., Fonseca, J.A.: "Using a planning scheduler to improve flexibility in real-time fieldbus networks", *IFAC Control Engineering Practice*, 7: 101-108, February 1999.
- [6] Martins, E., Neves, P., Fonseca, J.A.: "Architecture of a Fieldbus Message Scheduler Coprocessor Based on the Planning Paradigm", *Microprocessors and Microsystems*, Vol. 26, Issue 3, April 2002.
- [7] Pedreiras, P., Almeida, L., Gai, P., "The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency", to appear in *Proceedings of the 14th Euromicro Conference on Real-Time Systems*, Viena, Austria, June 2002.
- [8] Calha, M., Fonseca, J.A., "Adapting FTT-CAN for the joint dispatching of tasks and messages" submitted to *WFSC'2002 - IEEE Workshop on Factory Communication Systems*, Sweden, August 2002.
- [9] Pedreiras, P., Almeida, L., Fonseca, J.A.: "Improving the Responsiveness of the Synchronous Messaging System in FTT-CAN" *Proc. DCCS 2000: 16th IFAC Workshop on Distributed Computer Control Systems*, Sydney, Australia, 29 Nov. - 1 Dec. 2000.
- [10] Martins, E., Fonseca, J.A., "Improving Flexibility and Responsiveness in FTT-CAN with a Scheduling Coprocessor" *Proc of FET'2001 - 4th*

- FeT IFAC Conference Fieldbus Technology*, Nancy, France, 15-16 November, 2001.
- [11] Martins, E., Fonseca, J.A., “Traffic Scheduling Coprocessor with Schedulability Analysis Capability” *Proceedings of the Euromicro Workshop on Distributed Systems Design*, 2001.
- [12] Ferreira, J., P. Pedreiras, L. Almeida, J. Fonseca, “FTT-CAN Error Confinement”, *Proc of FET’2001 - 4th FeT IFAC Conference Fieldbus Technology*, Nancy, France, November 2001.
- [13] Ferreira, J., P. Pedreiras, L. Almeida, J. Fonseca, “The FTT-CAN protocol: improving flexibility in safety-critical systems”. in *IEEE MICRO special issue on Critical Embedded Automotive Networks*, July/August 2002.

AUTHOR(S)

Principal Author: José A. Fonseca holds a PhD in Electrical Engineering, Electronics, from the University of Aveiro, Portugal. At present he is an Associate Professor at the Department of Electronics and Telecommunications of the same University. He is also the co-ordinator of the distributed electronic systems team at the research unit IEETA – Instituto de Engenharia Electrónica e Telemática de Aveiro.

Co-authors: Joaquim Ferreira holds a MSc in Electronics and Telecommunications from the University of Aveiro, Portugal. He is a lecturer at Escola Superior de Tecnologia of the Polytechnic Institute of Castelo Branco in Portugal. He is currently a PhD student at the University of Aveiro, with a grant from PRODEP.

Mário Calha holds a MSc in Computer Science from the Instituto Superior Técnico, Lisbon. He is a lecturer at Escola Superior de Tecnologia of the Polytechnic Institute of Castelo Branco in Portugal. He is currently a PhD student at the University of Aveiro, with a grant from PRODEP.

Paulo Pedreiras holds a graduation in Electronics and Telecommunications Engineering from the University of Aveiro. He is currently a PhD student at the University of Aveiro with a grant from FCT (PRAXIS XXI).

Luís Almeida holds a PhD in Electrical Engineering, Electronics, from the University of Aveiro, Portugal. At present he is an Auxiliary Professor at the Department of Electronics and Telecommunications of the same University.

Presenter:

The paper is presented by José A. Fonseca



José A. Fonseca



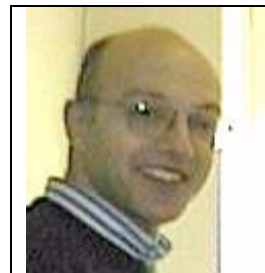
Joaquim Ferreira



Mário Calha



Paulo Pedreiras



Luís Almeida