EDF message scheduling on controller area network

by Paulo Pedreiras and Luís Almeida

The Controller Area Network has a maximum transmission rate of 1 Mbit/s and its fixed priorities-based medium access control limits the maximum bus utilisation when timeliness guarantees are required. An implementation of earliest deadline first (EDF) message scheduling on CAN, based on the FTT-CAN protocol, is presented, which allows higher utilisation factors with timeliness guarantees. The advantages of using EDF instead of rate-monotonic scheduling on FTT-CAN are highlighted, and a comparison with other implementations of EDF scheduling on CAN is presented.

The Controller Area Network (CAN)[1] protocol was developed in the mid 1980s by Robert Bosch GmbH, aiming at automotive applications, to provide a cost-effective communications bus for in-car electronics and as an alternative to expensive and cumbersome wiring looms. It is standardised as ISO 11898 for high-speed applications (1 Mbit/s) and ISO 11519-2 for lower speed applications (125 kbit/s). The maximum transmission rate that can be achieved is specified to 1 Mbit/s, but can be lower depending on the bus length and transceiver speed. However, there is a current trend across many application fields (e.g. automotive, machine tools, process and manufacturing industry), towards an increased number of interconnected devices, particularly intelligent sensors, and an increased amount of data to be shared. Therefore, the available bandwidth becomes scarce and thus message scheduling policies that maximise the utilisation factor while supporting timeliness guarantees are of special interest.

Two widely studied real-time scheduling algorithms are rate monotonic (RM) and earliest deadline first (EDF). Rate monotonic belongs to the class of static algorithms, in which scheduling decisions are based on fixed parameters assigned to tasks before their activation. EDF belongs to the class of dynamic algorithms, in which scheduling decisions are based on parameters that can change during system evolution. In RM scheduling, messages with shorter periods get higher priorities than messages with longer periods. In EDF, priorities increase as deadline expiry approaches.

In the context of real-time task scheduling in microprocessors, EDF seems more attractive than RM, since it allows full CPU utilisation to be obtained, whilst for RM the upper bound for guaranteed timeliness can be as low as 69%.[3] However, a simulation study carried out by Lehoczky, Sha and Ding[4] showed that RM is able to schedule random task sets with utilisation as high as 88%, on average. In the context of message scheduling, particularly on the CAN bus, some comparative results between RM and EDF using realistic loads[5] show a difference around 20% in network utilisation in favour of EDF.

Previous work on implementing EDF scheduling on CAN[6,7] relied exclusively on the native medium access control (MAC) of this protocol, in which messages are given static priorities depending on the message identifier. Since priorities in EDF are dynamic, this approach implies dividing the identifier in at least two fields: one to encode the (time-dependent) priority and another to identify the message itself. The major drawbacks of this approach are a reduction on the number of bits available for message identification, the need to perform periodically some processing to update

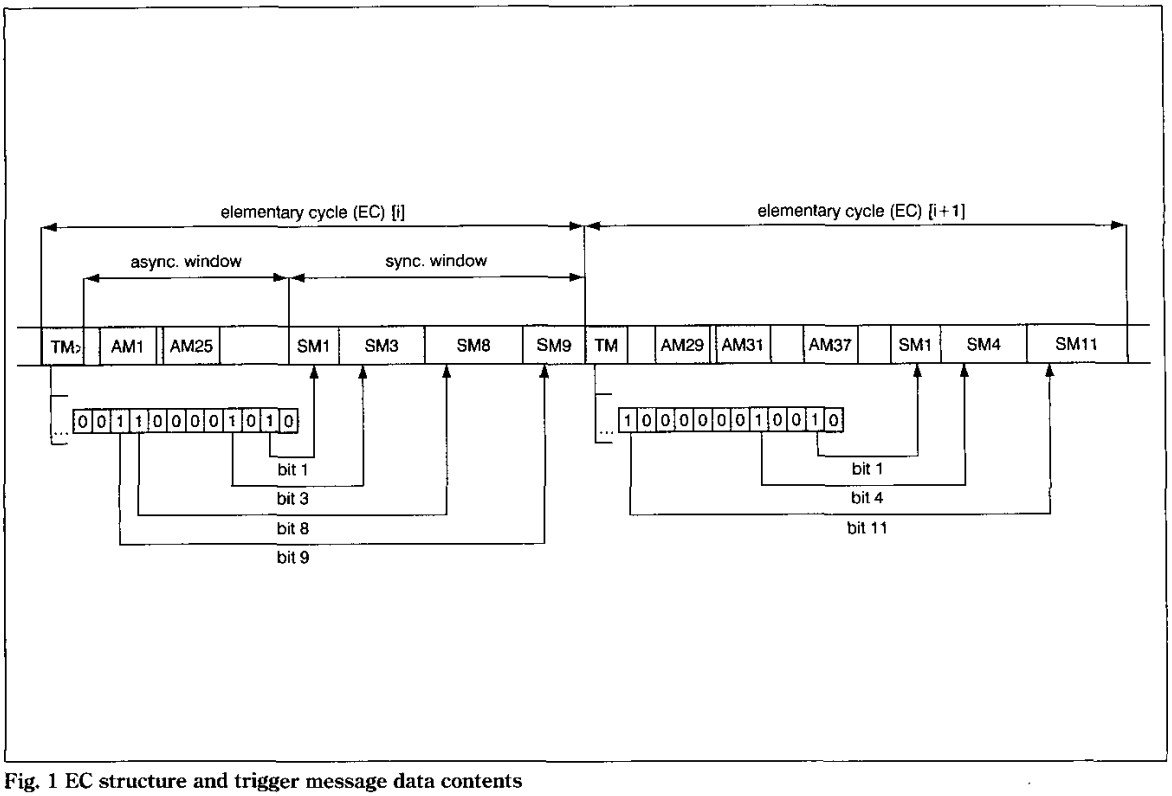

Fig. 1 EC structure and trigger message data contents

the dynamic priority field, the occurrence of priority inversions due to the limited resolution for expressing deadlines and the need to exchange time synchronisation messages.

In the FTT-CAN protocol (flexible time-triggered communication on CAN)[8] all transactions take place within fixed duration elementary cycles (ECs), supporting both synchronous and asynchronous traffic. The former type of traffic is scheduled centrally in a particular node called master. This node broadcasts one message that triggers the start of each EC and conveys the schedule of synchronous messages for that cycle. This technique has two relevant properties with respect to this work. Firstly, it allows the implementation of centralised scheduling with low communication overhead (typically between 2% and 10%, depending on the number of messages and bus speed). Secondly, it supports message scheduling policies that are independent of the one used at the native MAC of CAN, and furthermore, only the master node is aware of the particular type of scheduling being used.

This paper presents an implementation of EDF scheduling on CAN using the FTT-CAN protocol. It briefly presents the protocol and also simulation and experimental results highlighting the impact of using EDF and RM scheduling policies on FTT-CAN. It goes on to show a comparison with other proposals for EDF on CAN, and finally presents the main conclusions that can be drawn from this work.

FTT-CAN protocol

The FTT-CAN protocol has previously been presented in Reference 8. This protocol was designed to support the following properties:

- flexible handling of time-triggered traffic
- support for on-the-fly changes both on the message set and scheduling policy
- online admission control of change requests for the time-triggered traffic
- indication of temporal accuracy of time-triggered messages
- support for different types of traffic: time-triggered, event-triggered, hard real-time, soft real-time and non-real-time
- temporal isolation: time and event-triggered traffic do not disturb each other
- efficient use of network bandwidth.

In order to achieve these goals, the FTT-CAN protocol relies on centralised scheduling and master/multi-slave transmission control.

Centralised traffic scheduling allows both the communication requirements as well as the message

scheduling policy localised in one single node, the master, facilitating online changes to both. On the other hand, such centralisation also facilitates the implementation of online admission control in the master node.

Master/multi-slave transmission control allows the enforcement of the traffic timeliness in the bus without incurring a high penalty concerning the efficiency in bandwidth utilisation. The first aspect is typical of master-slave transmission control since the master explicitly tells each slave when to transmit, thus enforcing the traffic timeliness. The second aspect results from the fact that a single master message triggers the transmission of several messages by slave nodes.

In FTT-CAN traffic is allocated in fixed duration time slots called elementary cycles. The bus time is organised in an infinite succession of ECs. Each EC starts with a trigger message (TM), sent by the master, and is composed by two sequential phases for transmission of time and event-triggered traffic (Fig.1). The traffic on the synchronous window is controlled by the TM, which contains the identification, in a coded form, of the messages that must be transmitted within the respective EC (EC-schedule). The asynchronous window supports the transmission of event-triggered messages. This type of traffic relies on the native CAN arbitration mechanism and the FTT-CAN protocol confines to the asynchronous window the instants when these messages can effectively compete for bus access.

The synchronous traffic is time-triggered and it is controlled autonomously by the master node through the EC trigger message. Each node holds a table identifying which synchronous messages it produces. On reception of the EC trigger message, slave nodes decode the EC-schedule information and compare it with the table of the locally produced messages, in order to identify which synchronous messages it should produce in the current EC. These messages are then queued for transmission in the synchronous phase of the EC. Collisions on bus access between messages scheduled to be produced in a particular EC are resolved by the native distributed MAC protocol of CAN. All the mechanisms required to manage the transmission of synchronous messages are gathered in the so-called synchronous messaging system (SMS).

The FTT-CAN protocol also supports asynchronous traffic for event-triggered communication with external (application) control. The FTT-CAN protocol receives and queues the transmission requests originated in the application layer. After the beginning of the asynchronous window asynchronous messages are de-queued and placed in the transmission buffer to enter arbitration. In order to guarantee a temporal isolation between time and event-triggered traffic, the transmission of an asynchronous message is not allowed to extend beyond the duration of the respective window. This is enforced by removing from the transmission buffer any message that cannot be transmitted to

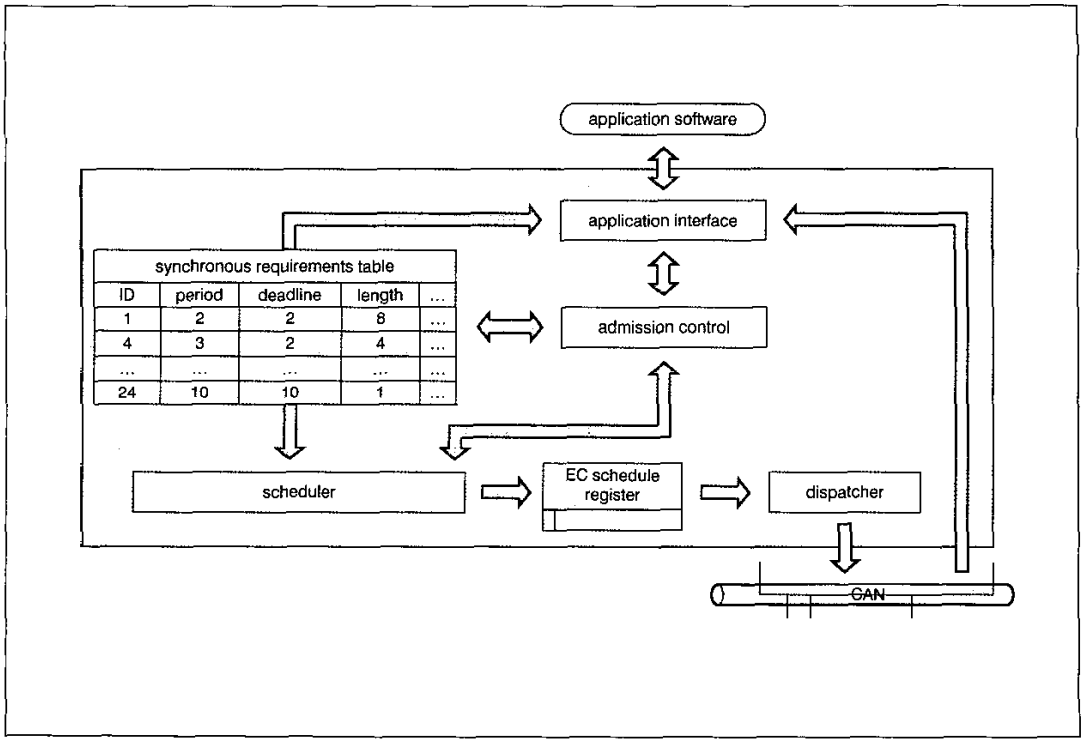

Fig. 2 Master node internal architecture

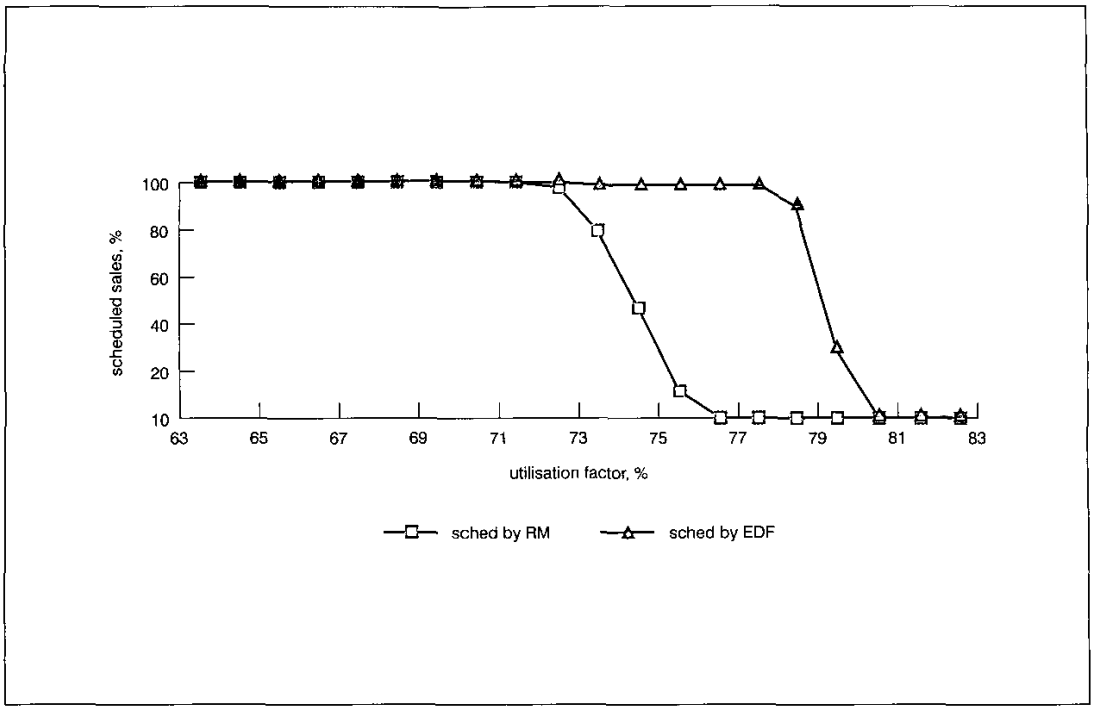

Fig. 3 Schedulability ratio by RM and EDF

completion before the end of the asynchronous window, and re-queuing it again in the next EC. The transmission of asynchronous messages is handled by the asynchronous messaging system (AMS).

The master node (Fig. 2) maintains a synchronous requirements table (SRT), which holds the properties of the synchronous messages, namely: identifier, period, relative deadline, initial phase, maximum transmission time and priority. Using this information the scheduler task selects which synchronous messages should be produced in each EC, according to the specific scheduling policy implemented. The identifiers of the messages that should be produced in each EC are stored in the EC-schedule register.

The dispatcher task is executed periodically and is responsible for reading the EC-schedule register, building the next EC trigger message with such EC-schedule and broadcasting it onto the network.

The application interface provides a set of services to manage the SRT, namely adding and removing messages as well as changing the properties of existing ones.

The FTT-CAN protocol allows online changes to the synchronous message set. Once a change request is received, the schedulability of the resulting message set is analysed. If it is not schedulable the request is denied. Conversely, if the resulting message set is schedulable, the SRT is updated accordingly.

As stated above, in FTT-CAN the synchronous traffic is scheduled centrally, on the master node, and it does not rely on the message's CAN identifiers. Rather, any message property, such as periodicity, deadline or importance, can be used. This makes it possible to obtain behaviours that are not possible with native CAN, e.g. dynamic priorities, fairness among different message streams requiring similar quality of service, robust scheduling etc. Moreover, slave nodes are unaware of the particular scheduling policy used, since they strictly follow the EC-schedules broadcast through the TM. Thus, the complexity of the particular scheduling policy implemented affects only the master node.

EDF and RM scheduling on FTT-CAN

This section presents several experimental and simulation results concerning the implementation of both RM and EDF on FTT-CAN. The results allow the verification of the gains in level of schedulability when EDF is used instead of RM, as well as the impact that these scheduling policies have on network-induced jitter and delay.

Computational overhead

As mentioned in the previous section, the scheduler in FTT-CAN is implemented centrally on the master node, therefore none of the producer/consumer nodes on the system is aware of the scheduling policy being used. The main consequences of this approach are:

- extra workload resulting from using EDF instead of

RM exclusively affects the master
- no updates are required to the software running on the producer and consumer nodes, whenever the message scheduling policy is changed, e.g. in the course of a transient overload.

Level of schedulability

In this section several simulation results are shown concerning the schedulability of random message sets in FTT-CAN under RM and EDF. The purpose is to assess the advantage of using EDF in the scope of FTT-CAN with respect to the level of schedulability. Before presenting the results, though, several practical aspects must be accounted for.

In FTT-CAN each EC starts with the transmission of an EC trigger message (TM). The length of this message, also set at pre-run-time, depends on the maximum number of synchronous messages in the SRT, i.e. the number of entries in the SRT. The following results have considered a maximum number of 32 synchronous messages; therefore using an EC trigger message with 4 data bytes yields a maximum length (LTM) of 95 bits, stuff bits included.

The overheads for reception and decoding of the EC trigger message, as well as queuing of the messages for transmission, must also be accounted for. In the particular case of the hardware test platform described in Reference 8, based on 8051 microcontrollers, these overheads (POVRHEAD) have been experimentally determined. An upper bound of 1ms (roughly 120 bits at 123 kbit/s) was found. It should be stressed that this factor is not a protocol overhead, but instead an implementation issue strongly related to the processing power available. For hardware platforms with higher processing capacity this factor can be a small fraction of the transmission of the trigger message, and therefore its impact becomes marginal. The maximum length of the synchronous phase (LSW) is computed by subtracting from the EC duration (LEC) the length of the trigger message (LTM), the TM reception and decoding overheads (POVRHEAD) and a possible minimum duration guaranteed for the asynchronous phase (LAW). The resulting expression is:

$$LSW = LEC - (LTM + POVRHEAD + LAW)$$

Considering that no minimum bandwidth is reserved for asynchronous traffic (i.e. LAW = 0), an EC duration of 8·9 ms and a transmission rate of 123 kbit/s, the maximum duration for the synchronous phase (LSW) is around 7·1 ms (approximately 80% of the EC duration).

In order to assess the actual difference in scheduling capability between RM and EDF in FTT-CAN, a simulation with 10 000 random message sets was performed. Each set had 32 messages with the following constraints:

- 5 messages with period 1 EC
- 10 messages with period between 3 and 6 ECs uniformly distributed
- 17 messages with period between 10 and 16 ECs uniformly distributed
- data length: 1-8 bytes uniformly distributed
- IDs are ordered by increasing period.

The purpose of using this pattern is to obtain sets with high network utilisation and with message periods in three different ranges. Fig. 3 shows how many of these sets are schedulable by RM and by EDF.

It can be observed that all sets in the simulation with utilisation factor up to 71% are schedulable both by RM and EDF, and those with utilisation between 71% and 77% are all schedulable by EDF but not all by RM. As can be observed in Fig. 3, EDF practically allows full utilisation of the fraction of the EC reserved for synchronous traffic (80%). The small reduction in schedulability (77%) is due to the inserted idle-time technique used to assure that this type of traffic never extends beyond the synchronous window of each EC (see earlier section on FTT-CAN protocol).

Network-induced jitter

Different scheduling policies generate different message schedules, exhibiting different properties concerning relevant parameters such as jitter, lateness etc.[2] For real-time control systems these properties can be particularly relevant. Namely, for many control applications (e.g. machine tools with numerical control, high speed servoing), jitter might have a strong impact on control quality.

To evaluate the impact of RM and EDF scheduling on message transmission jitter, a simulation was carried out using 10 000 random message sets following the same pattern as in the previous section. In the remainder of this discussion, the parameters are defined as follows:[2]

- $r_{i,j}$: release time of the jth instance of message m_i
- $s_{i,j}$: start time of the jth instance of message m_i
- $RJ_i = \max_j((s_{i,j} - r_{i,j}) - (s_{i,j-1} - r_{i,j-1}))$: relative release jitter of message m_i
- $AJ_i = \max_j(s_{i,j} - r_{i,j}) - \min_j(s_{i,j} - r_{i,j})$: absolute release jitter of message m_i.

Fig. 4 plots the absolute release jitter for both RM and EDF scheduling. The jitter values on the vertical axis are relative to the message periods.

It can be observed that the maximum absolute release jitter for messages with higher ID (lower priority for RM) is above 90% for RM, while for EDF it is at most 80%. This can be explained by the fact that for RM the messages with higher ID have lower priority than the ones with lower ID, and thus are always scheduled last. However, for EDF, the priority is dynamic and increases as the deadline approaches, therefore higher ID messages that are successively delayed can get a priority higher

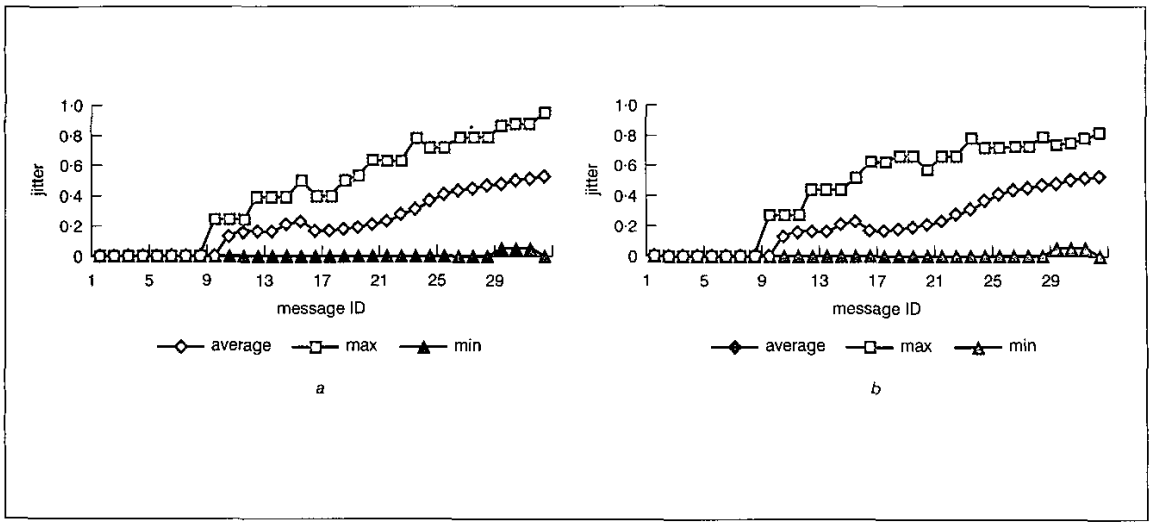

Fig. 4 Absolute release jitter using (*a*) RM and (*b*) EDF scheduling algorithms

than messages with lower ID. The trade-off is an increase on the absolute release jitter suffered by the intermediate ID messages under EDF. A similar but less pronounced behaviour can be observed for the average absolute release jitter.

Fig. 5 shows the relative release jitter for the same sets of messages. As can be observed, the overall effect of changing the scheduling policy from RM to EDF is consistent with the observations made above for the case of absolute release jitter.

Therefore, using EDF instead of RM allowed the reduction of jitter figures for messages with longer periods (higher ID) at the expense of a slight increase in these figures for the messages with intermediate periods.

Comparison of EDF on FTT-CAN with other proposals for EDF on CAN

Previous work on implementing EDF scheduling on CAN[6,7] relied exclusively on the native MAC of the protocol. Since the priority of the messages depends on the identifier bits, and priorities in EDF are dynamic, this approach implies dividing the identifier into at least two fields: one to encode the priority and another to identify the message itself. Several techniques for managing the priority field are discussed in the literature,[6,7] which consider the restriction of using a limited number of identifier bits as well as the need to keep the processor overhead in acceptable levels. A possible solution[6] is based on the encoding of absolute deadlines relative to a periodically increasing time reference designated *epoch*. However, this solution has difficulties in dealing with message sets containing periods that are orders of magnitude apart. In this case either a coarse time granularity is used, leading to a large number of priority inversions, or the number of bits used to encode the deadline is increased, reducing the number of distinct messages that can be scheduled. In Reference 7 the author proposes encoding the time to the absolute deadline in a logarithmic time scale, increasing the temporal resolution as deadlines are approached and thus reducing the number of possible priority inversions for early deadlines. A consequence of this technique is that the identifier bits, used to encode the priority of the messages waiting for transmission, must be updated each time messages compete for the bus access after it becomes idle (referred to as *arbitration round*). Major drawbacks of these approaches have been already referred to earlier in this paper, and can be summarised as:

- support for fewer messages due to using several identification bits to encode the priority
- higher processing requirements in all nodes to periodically update the priority field
- difficulties in handling messages with a wide range of relative deadlines.

As opposed to these approaches to EDF scheduling on CAN, in the FTT-CAN protocol all the scheduling decisions are performed in the master node. Consequently, none of the drawbacks presented above hold.

Firstly, in FTT-CAN the priority, i.e. time to the deadline in the case of EDF, is held in a variable within a data structure and no identifier bits are used to encode it. Thus, no reduction is imposed on the number of messages.

Secondly, the scheduling activity is confined to the master. The EC trigger message identifies the synchronous messages that must be produced in each EC. All other nodes follow a slave-like operation that is completely independent of the scheduling technique used by the master. Thus, the use of EDF does not impose any extra computational activity in any node beyond the master.

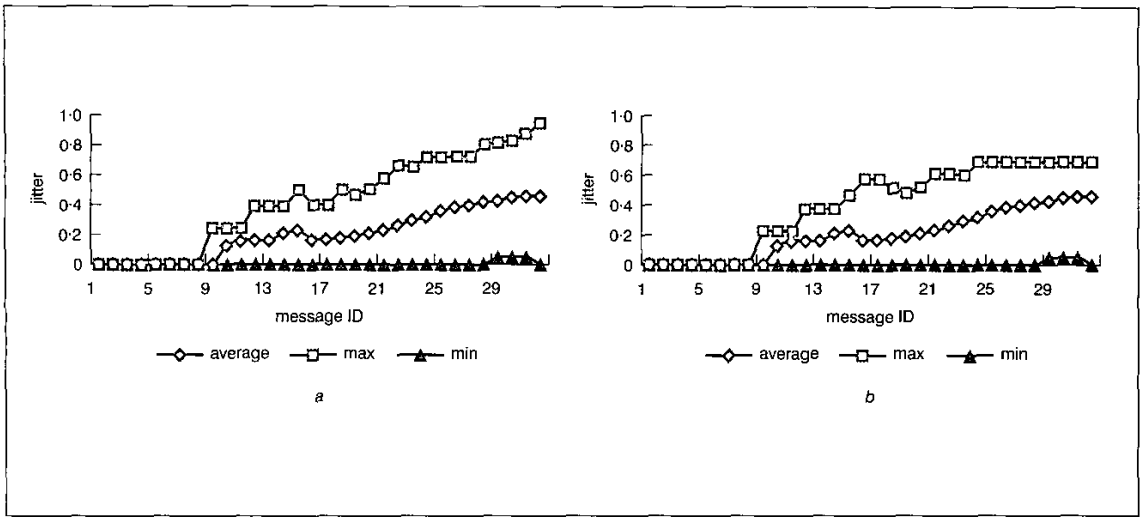

Fig. 5 Relative release jitter using (*a*) RM and (*b*) EDF scheduling algorithms

Finally, the SRT is maintained in an adequate structure in the master memory. Message parameters, such as periods and deadlines, are held within variables, the type of which can be chosen appropriately to support a required range of values. Thus, the range of periods that can be handled within FTT-CAN is virtually unlimited, beyond the constraint of being integer multiples of the EC duration.

Some results presented in the literature[6,7] are optimistic, since all nodes are assumed to have synchronised clocks, but the load generated by a clock synchronisation protocol is not included. Despite this, to allow a global comparison between these methods and the one proposed in this paper, a simulation was performed under similar conditions as referred to by Di Natale,[7] namely using random message sets with 30 messages grouped in three distinct categories according to their periods, 3-12 ms, 30-120 ms and 250 ms-1 s. The deadline to period ratio is in the range 0·8-1 uniformly distributed. Concerning the protocol configuration, the EC was set to 3 ms, the transmission rate to 250 kbit/s, no asynchronous window, and finally reception and decoding overheads (POVRHEAD) of 1 ms. The results obtained are plotted in Fig. 6. For each point in the plot 5000 random sets were generated, giving a total of 60 000 message sets. To allow an easier comparison with the results obtained by Di

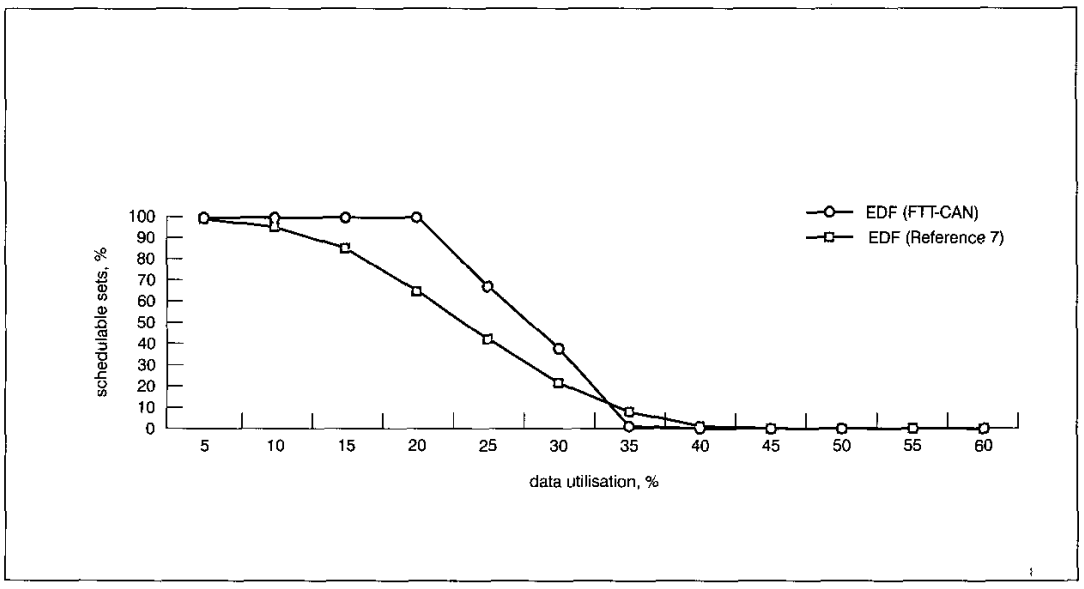

Fig.6 Percentage of schedulable sets using EDF on FTT-CAN

Natale,[7] the x axis shows the effective data utilisation, i.e. equivalent transmission time of data bits only, over the message period.

Comparing both results, with FTT-CAN the curve is more abrupt, presenting a larger level of schedulability for a wider range of data utilisation values.

When compared to the approaches based on ID field manipulation, the FTT-CAN based EDF scheduling implementation presents the following advantages:

- Simplicity of scheduler implementation in the master node. Furthermore, the scheduling policy can easily be changed online, e.g. during transient overloads.
- Message scheduling separated from the MAC arbitration, avoiding the undesirable compromise between dynamic priorities and message identifiers.
- CPU load required by EDF scheduling confined to the master. Remaining nodes require a constant CPU load to decode the EC trigger message, whichever is the scheduling policy being used.
- Support for virtually unlimited range of messages' periods and deadlines by using appropriate types for the respective variables.
- Built-in online admission control, providing timeliness guarantees on message delivery, even for dynamic message sets.

However, the FTT-CAN implementation also presents some drawbacks. On the one hand there is a limitation imposed on the temporal resolution. As stated earlier in the paper, in FTT-CAN all periods and deadlines are expressed as integer multiples of the EC duration and a sub-EC resolution is not supported. This limitation, nevertheless, does not seem to be particularly relevant since for typical applications (e.g. automotive, machine tool control) the shortest deadlines and periods lie in the range 1-10 ms, which is the same magnitude of the envisaged EC duration in FTT-CAN systems. On the other hand, the protocol overhead depends on the maximum number of synchronous messages, since each distinct message requires the exclusive use of one bit in the trigger message data field. Finally, EDF scheduling in FTT-CAN is only supported with respect to the synchronous messages. Asynchronous messages are transmitted according to CAN fixed priorities.

Conclusion

This paper has presented an implementation of earliest deadline first message scheduling on the Controller Area Network based on the FTT-CAN protocol (flexible time-triggered communication on CAN). In order to clearly establish the advantages of using EDF instead of fixed priorities scheduling such as rate monotonic, a comparison between RM and EDF in terms of network-induced jitter, release delay and level of schedulability has been presented. Simulation results show not only that EDF does allow a greater bus utilisation factor to be obtained than RM under guaranteed timeliness, but also that it allows jitter and release delays for messages with longer periods to be reduced. The major drawback is an increase in the CPU load, which in this case is felt at the master node only. This fact brings along another advantage, that is the ease of changing the scheduling policy online. For EDF scheduling, such a possibility is particularly relevant when the system is expected to react to transient overloads.

A comparison between EDF on FTT-CAN and previous works on EDF on CAN is also presented. By comparing simulation results, it can be concluded that the approach presented in this paper allows similar results to be obtained for bandwidth utilisation by synchronous messages. However, unlike the other approaches, the extra complexity required to implement EDF is only reflected on one node, i.e. the master. Furthermore, since the scheduling does not rely on the CAN identifier field, the addressing scheme is not compromised and wide ranges of relative deadlines are easily supported. Finally, built-in online admission control delivers timeliness guarantees with dynamic message sets. The main trade-off is a limited temporal resolution, since message deadlines are constrained to be integer multiples of the EC duration.

Acknowledgments

An extended version of this paper was presented at the IEEE/IEE Real-Time Embedded Systems Workshop, London, December 2001.

This work was partially supported by the Portuguese Government through grant PRAXIS XXI/BD/21679/99.

References

1 CAN Specification Version 2.0: Robert Bosch GmbH, Stuttgard, 1991
2 BUTTAZZO, G.: 'Hard real-time computing systems—predictable scheduling algorithms and applications' (Kluwer Academic Publishers, 1997)
3 LIU, C., and LAYLAND, J.: 'Scheduling algorithms for multiprogramming in a hard real-time environment', *Journal of ACM*, 1973, **20**, (1), pp. 46-61
4 LEHOCZKY, J., SHA, L., and DING, Y.: 'The rate monotonic scheduling algorithm: exact characterisation and average case behaviour', Proceedings of the IEEE Real-Time Systems Symposium, 1989
5 ZUBERI, KHAWAR, and KANG SHIN: 'Scheduling messages on Controller Area Network for real-time CIM applications', *IEEE Transactions on Robotics and Automation*, 1997, 13
6 ZUBERI, KHAWAR, and KANG SHIN: 'Non-pre-emptive scheduling of messages on Controller Area Network for real-time control applications', Proceedings of Real-Time Technology and Applications Symposium, 1995
7 DI NATALE, M.: 'Scheduling the CAN bus with earliest deadline techniques', Proceedings of the IEEE Real-Time Systems Symposium. 2000
8 ALMEIDA, L., FONSECA, J., and FONSECA, P.: 'Flexible time-triggered communication on a Controller Area Network', Proceedings of the Work-In-Progress Session of RTSS '98 (19th IEEE Real-Time Systems Symposium), Madrid, Spain, December 1998

The authors are with DET-IEETA, Universidade de Aveiro, P-3810-193 Aveiro, Portugal; pedreiras@alunos.det.ua.pt; lda@det.ua.pt