PERGAMON

Control Engineering Practice 7 (1999) 1515–1523

www.elsevier.com/locate/conengprac

Scheduling hard and soft real-time communication in a controller area network

Mohammad Ali Livani[a,]*, Jörg Kaiser[a], Weijia Jia[b]

[a]University of Ulm, Department of Computer Structures, 89069 Ulm, Germany
[b]City University of Hong Kong, Department of Computer Science, Hong Kong

Received 10 February 1999; accepted 15 July 1999

Abstract

The paper introduces a mechanism to implement distributed scheduling for the CAN-bus resource in order to meet the requirements of a dynamic distributed real-time system. The key issues considered here are multicasting, distinguishing between hard real-time, soft real-time, and non-real-time constraints, achieving high resource utilization for the CAN-bus, and supporting dynamic hard real-time computing by allowing dynamic reservation of communication resources. © *1999 Elsevier Science Ltd. All rights reserved.*

Keywords: Scheduling; Fieldbus; Timing constraints; Real-time communication; Periodic and aperiodic traffic; Distributed algorithms; CAN

1. Introduction

A real-time communication system (RTCS) constitutes the backbone of distributed control applications. In order to support the timely completion of distributed real-time activities, RTCS must ensure bounded delivery delay for real-time messages. Assuming the existence of hard real-time, soft real-time and non-real-time-distributed activities in a complex distributed real-time system, communication activities can also be considered to have hard real-time, soft real-time, and non-real-time natures. While the timeliness of hard real-time communication must be guaranteed, the delivery delay of a soft real-time message may exceed the desired bound in overload situations. For the delivery of non-real-time messages no delay bound is defined. This behavior of the real-time communication must be provided to the application level by the RTCS.

The design of a real-time communication system must exploit the predictability features of the lower system levels, in order to achieve predictable behavior at the higher levels. If a communication medium exhibits unpredictable behavior (e.g. unbounded transmission delay) then a high-level communication protocol will hardly be able to provide a predictable real-time communication to the application.

In the area of industrial automation and the automotive industry, field busses are used to disseminate time critical messages because of their high predictability. Among field busses, the CAN bus (BOSCH, 1991) provides advanced built-in features, which make it suitable for complex real-time applications. Some of these features include high robustness against electro-magnetic interference, priority-based multi-party bus access control, bounded message length, efficient implementation of acknowledgment and error indication, and automatic fail-silence enforcement with different fault levels.

As a common resource, the CAN bus has to be shared by all computing nodes. Access to the bus has to be scheduled in a way that allows distributed computations to meet their deadlines in spite of competition for the communication medium. Since the scheduling of the bus cannot be based on local decisions, a distributed consensus about the bus access has to be achieved.

There exist several alternative approaches to solve this problem. All of them take advantage of the priority-based arbitration mechanism of CAN (cf. Section 3), which is also called carrier sense multiple access with collision avoidance (CSMA/CA). Unlike the CSMA/CA mechanism known from the wireless LAN literature, this mechanism does not avoid collisions; it uses collisions to

*Corresponding author. Tel.: +49-731-502-4177; fax: +49-731-502-4182.
E-mail address: mohammad@informatik.uni-ulm.de (M. Ali Livani)

compare frame identifiers and assess the frame with the lowest identifier value on the fly.

Deadline-monotonic priority assignment (Audsley, Burns, Richardson & Wellings, 1992; Tindell & Burns, 1994) ensures that deadlines are met by an off-line feasibility test for a static system with periodic and sporadic tasks. Due to the tight relationship between the identifier and the priority of CAN messages, the fixed priority assignment has been applied in the most common CAN-based communication systems implicitly, e.g. CAL (CiA, 1993), SDS (Crovella, 1994), and DeviceNet (Noonen, Siegel and Maloney, 1994).

The CANopen standard (CiA, 1996), which is based on CAL, defines periodic communication. According to this scheme, any synchronous message is transmitted with a high priority at most once during each "communication cycle period". The transmission of synchronous messages usually takes only part of a communication cycle. During the rest of the communication cycle, asynchronous messages — which do not have a well-defined occurrence cycle — are transmitted with lower priorities. The CANopen standard approaches the determinism of time-division multiple-access (TDMA) using the fixed priority scheme. However, while a real TDMA protocol like the time-triggered protocol (TTP) (Kopetz & Grünsteidl, 1994) is well suited for fault-tolerant applications, CANopen systems are generally not fault-tolerant. This is because in CANopen a communication cycle begins with the propagation of a special message (the SYNC object) by a certain node (the SYNC master). Thus, the SYNC master constitutes a single point of failure for the whole system.

In Zuberi and Shin (1995) a combination of fixed and dynamic priority scheduling is adopted. But this approach fails to schedule messages in a bus with three or more sender nodes because of too short a *time horizon*. The term *time horizon* is explained later in the paper.

This paper presents a hybrid bus scheduling algorithm that combines the determinism of TDMA and the flexibility of dynamic Least-Laxity-First resource scheduling.[1] The algorithm provides a mechanism to reserve enough resources for critical hard real-time messages in order to guarantee their timely transmission under anticipated fault conditions. Even in situations where overload is caused by a burst of soft real-time messages, deadlines of hard real-time messages are met.

Unlike static TDMA schemes like TTP (Kopetz & Grünsteidl, 1994), the proposed scheme allows for the dynamic reservation of communication resources for tasks, which only start if they obtain a guarantee of timeliness (Stankovic & Ramamritham, 1989). Another difference between hybrid bus scheduling and TDMA is that in the static TDMA approach the reserved resources are either used by their 'owner' or they are left unused. In contrast, in the hybrid bus scheduling approach, soft real-time messages compete for unused reserved resources according to the LLF scheme. In a fault-tolerant system where time redundancy is applied to enable communication failures to be tolerated, the reserved redundant resources are usually not needed by the owner. Thus the hybrid scheduling mechanism achieves a higher resource utilization by reusing redundant reserved times for non-critical communication. Best-effort service of soft real-time communication is achieved by applying non-preemptive LLF scheduling, which is comparable to optimal EDF scheduling, and by assigning low priorities to non real-time messages.

The paper is organized as follows: Section 3 gives a brief description of the CAN arbitration mechanism. Section 4 illustrates the underlying system model. Section 5 presents the hybrid bus scheduling mechanism, which is a combination of dynamic TDMA scheduling for hard real-time communication and the LLF scheduling for soft real-time communication, both of which were designed on top of a dynamic priority scheme. A summary concludes the paper.

2. Notation

Network parameters

f	is the anticipated maximum number of consecutive transmission failures,
$\lambda_{\max}$	the anticipated maximum bit-error rate of the bus,
Δbt	the nominal bit-time of the bus, e.g. 1 μs at 1 Mbit/s,
ΔO_c	the maximum offset between the global time reference and any of the synchronized local time references in the network,

Message parameters

b_m	is the number of data bytes in a message m,
L_m	the worst-case length of a message m in bits under assumption of maximum bit-stuffing ($= b_m \times 9.6 + 75$),
d_m	the transmission deadline of a message m, i.e. the time when m must be transmitted successfully.
ΔT_m	the worst-case transmission duration of a message m, i.e. $\Delta bt L_m$,
LST_m	the latest transmission start time of a message m, i.e. $d_m - \Delta T_m$,
ΔL_m	the transmission laxity of a message m, i.e. the time until LST_m,

[1] Due to the non-preemptive nature of the CAN bus and the low variability of the length of CAN frames, the LLF and EDF (earliest deadline first) schemes perform similarly here. However, LLF was preferred for implementation reasons.

$\Delta T_{\max}$	the worst-case transmission duration of any arbitrary message ($= 151 \times \Delta bt$),
ΔT_{block}	the worst-case blocking time due to the non-preemptive transmission of any arbitrary message ($= 154 \times \Delta bt$),
ΔT_{fail}	the worst-case time loss due to the transmission failure of any arbitrary message ($= 168 \times \Delta bt$).

Scheduling parameters

ΔR_m	is the length of reserved time slot for a hard real-time message m,
S_m	the beginning of the reserved time slot of a hard real-time message m,
LRT_m	the latest ready time of a hard real-time message m,
$\Delta R_{\min}$	the length of the shortest reserved time slot in the system,
$\Delta R_{\max}$	the length of the longest reserved time slot in the system,
$\Delta G_{\min}$	the minimum gap between two different reserved time slots, which is required due to inaccurate global time reference ($\Delta G_{\min} = 2 \times \Delta O_c$),
$P_H(\Delta L)$	the priority value for a hard real-time message with the current laxity ΔL,
$P^{\text{HRT}}_{\min}$	the minimum priority value (highest possible priority) for any hard real-time message,
$P^{\text{HRT}}_{\max}$	the maximum priority value (lowest possible priority) for any hard real-time message,
$P_S(\Delta L)$	the priority value for a soft real-time message with the current laxity ΔL,
$P^{\text{SRT}}_{\min}$	the minimum priority value (highest possible priority) for any soft real-time message,
$P^{\text{SRT}}_{\max}$	the maximum priority value (lowest possible priority) for any soft real-time message,
$P^{\text{NRT}}_{\min}$	the minimum priority value (highest possible priority) for any non real-time message,
$P^{\text{NRT}}_{\max}$	the maximum priority value (lowest possible priority) for any non real-time message,
Δt_p	the priority slot: the portion of time, where a dynamic priority remains unchanged,
ΔH	the time horizon: the amount of future time, which is visible to an EDF or LLF scheduler.

3. CAN arbitration mechanism

The key to understanding the CAN-Bus is the fact that all nodes scan the value of every bit while it is being transmitted. Hence, all correct nodes have a consistent view of every bit. CAN controllers are commonly connected by a wired AND circuit, i.e. whenever different bit values are sent by different nodes simultaneously, the logical AND function of the bit values is observed by all nodes (including the senders).

This consistent global view is exploited for a priority-based arbitration mechanism. According to this mechanism, as soon as the bus is idle, each node competing for the bus begins to send the arbitration field of its message, which mainly consists of an 11- or 29-bit identifier (depending on the "standard" or "extended" format — Fig. 1). If at this time a node sends a '1' and senses a '0', it becomes aware of a collision, stops transmitting, and switches to the receiver mode. At the end of the arbitration field, only the node which is sending the message with the lowest arbitration field value, will be transmitting.

Due to this arbitration technique, the identifier of a CAN-message serves as its priority, and the bus acts as a priority-based dispatcher.

The priority-based arbitration mechanism requires that different CAN nodes never simultaneously send messages with equal identifiers. This requirement must be satisfied by the communication software.

4. The system model

The authors assume a number of different micro-controllers with different performance attributes ranging from 8- to 32-bit architectures. All micro-controllers are equipped with a CAN–Bus interface. The model exploits the inexpensive availability of 8-bit micro-controllers to structure the overall task of the system into small packages, i.e. objects with a well-defined message interface to other objects. Of course, it is possible to locate several tightly related objects in one node.

Higher control instances are available to control groups of objects or eventually to control the entire system using more powerful micro-processors. Thus, the

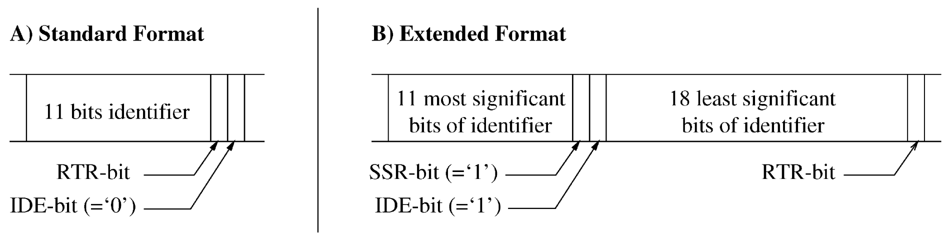

Fig. 1. The CAN arbitration field.

higher control instances have a well-defined instrumentation interface composed of objects, each encapsulating a certain functionality.

4.1. Object groups

In conventional object-oriented programming, an invocation is directed at a single object blocking the client like a procedure call. It is usually not possible to express a request to a group of objects. However, in a real-time control system it is beneficial to provide groups of objects and to use multicasts to invoke these groups. Group invocation enables simple solution of problems like object replication to achieve fault-tolerance, or the coordination of synchronous distributed activities.

In this system model, an object participating in group communication does not necessarily have any knowledge about the number and location of the other group members. Therefore, the sender of a multicast usually does not know whether it has to send the message locally or remotely. This simplifies the design and implementation of the objects, and minimizes the configuration effort when adding or removing an object.

But consequently, there must be an object in the system that knows the configuration. The configuration is given by multicast receiver groups, where a group contains one or more objects. This configuration information is maintained by distributed multicast agents, one residing on every computing node. Every object wishing to send a multicast message requests its local multicast agent to deliver the message to the group members. The multicast agent is described in more details in Kaiser and Livani (1998).

4.2. Group communication protocol

In order to support real-time object groups with consistent information, the group communication protocol must deliver real-time messages to all members of a group in a *timely* and *consistent order*. Section 5 describes how to guarantee timely multicast transmission in CAN. Given the guarantee of timely message transmission, (Livani & Kaiser, 1999) have shown that atomic multicast can be achieved by consistent ordering of messages based on application-defined delivery deadlines.

4.3. The global schedule with different urgency classes

In a complex real-time system, such as an attack helicopter, different distributed computations with hard and soft deadlines have to be performed. There are periodic control tasks which are necessary for the aircraft to continue flying, and sporadic pilot commands, which must also be considered as having hard deadlines. For such tasks, adequate resources must be reserved in order to guarantee their timely completion. In contrast, functions like target tracking may cause an arbitrarily high load on the computing system depending on the environmental conditions. Although timely execution of these activities is generally important for the system, it is not possible to reasonably guarantee hard deadlines for them.

Another application area with a mixed load of hard and soft real-time activities, is drive-by-wire. Periodic hard real-time activities like motor management and by-wire steering must be performed periodically, and critical sporadic activities like anti-lock brake control or electronic stabilization programs (ESP) have to be guaranteed at their highest occurrence rates. But activities like gear changes are less critical, and may be delayed under certain circumstances, e.g. when the electronic stabilization program is activated.

Thus there are distributed computations which, however important, must be performed as best as possible without guarantee, and their deadlines are considered soft. These computations must use non-reserved resources. If sporadic service requests with hard deadlines do not occur at their highest anticipated frequency, then some reserved resources remain idle, and can also be used for soft real-time computation. If fewer errors occur than anticipated by the system's design, then probably some redundant reserved resources for hard real-time activities remain unused, and may also be used for soft real-time activities. To optimally utilize the system resources for soft real-time distributed computations while guaranteeing hard deadlines, a hybrid scheduling mechanism must be applied, which features resource reservation as well as dynamic deadline scheduling.

Beside real-time activities, there may also be non-real-time activities, which may only use resources that are not requested by real-time activities.

The global scheduling in a distributed system requires consensus between all participants about the use of shared system resources. The global schedule has to be enforced by all components, based on their local information. In a completely static system, a periodic global calendar is available and each component has its relevant entries referring to its activities in a global time scale. A global activity may only be started according to this schedule (Kopetz & Merker, 1985; Kopetz & Grünsteidl, 1994). In a more dynamic system where hard real-time, soft real-time, and non-real-time tasks coexist, things are more complicated. If a computing resource is free, a less critical (soft or non-real-time) task may start computation and request resources. In this case, it must not lead to the timing failure of a hard real-time activity.

The scheduling of the network is a central part of the global resource scheduling problem. In the next section, a hybrid global scheduling approach for the CAN bus is described, which is appropriate for application systems with hard and soft real-time distributed activities. The hybrid scheduling mechanism is mainly based on a dynamic priority scheme, and resource reservation.

5. The hybrid scheduling mechanism

5.1. The dynamic priority scheme

The dynamic priority scheme of the hybrid bus scheduling algorithm implements the least-laxity-first scheme. In order to realize LLF in a CAN network, the mapping of the transmission laxity into the message priority has to be defined, so that a message with a shorter transmission laxity wins the bus arbitration against a message with a longer laxity. In CAN the entire identifier is used as a priority field. However, the whole identifier cannot be used as a deadline-driven dynamic priority because of following reasons: Firstly, a subject-related message name should be encoded into the identifier in order to support subject-based addressing and exploit the message filtering features of the CAN controller hardware. Secondly, the sender node identifier must be included into the identifier field of CAN messages, to ensure that competing messages always have different identifiers. Note that this uniqueness cannot be achieved by subject-related unique naming of messages in a system with replicated objects. Fig. 2 illustrates how the CAN identifier is partitioned in order to fulfill all requirements mentioned above.

In the priority field of a real-time message, the time remaining until its latest transmission starts (let us call it *transmission laxity*) is encoded. Choosing the LLF scheme is justified by the following facts: firstly, the local scheduling decision on each node can be simplified by a queue of outgoing messages sorted by the transmission laxity. Secondly, a laxity can easily be transformed into a unified scale of priorities. Thirdly, the global priority-based message dispatching is performed by the arbitration mechanism of CAN at no cost. Due to the fact that the lower binary values of the priority field represent higher priorities, a shorter (lower) laxity is mapped to a lower value in the priority field.

Having a fixed value range $\{P_{\min} \cdots P_{\max}\}$ for the priority field, a transmission laxity value ΔL is mapped to a priority value P, where $P = \lfloor \Delta L / \Delta t_p \rfloor + P_{\min}$ for $\Delta L < (P_{\max} - P_{\min})\Delta t_p$ and $P = P_{\max}$ for $\Delta L \geq (P_{\max} - P_{\min})\Delta t_p$. The modification of the dynamic priorities can be performed by periodically decreasing P once per Δt_p.

8 bits	8 bits	13 bits
Priority	TxNode	Message name

Fig. 2. Partitioning of a CAN-message identifier.

The period Δt_p is called the priority slot. Since all laxity values $\Delta L \geq (P_{\max} - P_{\min})\Delta t_p$ are mapped to the same priority, the priority-based dispatcher (i.e. the CAN arbitration mechanism) cannot distinguish different laxity values which are greater or equal $(P_{\max} - P_{\min})\Delta t_p$. Hence $(P_{\max} - P_{\min})\Delta t_p$ is the *time horizon* of the proposed LLF-scheduler. The term *time horizon* denotes the amount of time in the future that can be correctly analyzed and planned by a time-based decision-maker, like an EDF or LLF scheduler.

The required time horizon of a scheduling mechanism depends on the timing requirements of the real-time application. In the following discussion, a CAN bus is assumed, where in worst case N transmitters attempt to send messages with equal laxity, and another one has a message with a larger laxity. If all laxity values are larger than the time horizon of the LLF scheduler, then they will be transformed into the same priority $P_{\max}$. Then if the message with the larger laxity wins the arbitration, and the remaining time is too short to schedule the other N messages after it, then the scheduler will fail because of too short a time horizon. In order to avoid a scheduling failure in this case, the time horizon should not be shorter than the time required for the transmission of an arbitrary set of N messages.

Assuming a maximum transmission time of $\Delta T_{\max}$, an inter-frame spacing of 3 bits, a maximum transmission failure rate of $\lambda_{\max}$, and a maximum time loss of ΔT_{fail} per transmission failure, the time horizon ΔH must satisfy the condition: $\Delta H \geq N(\Delta T_{\max} + 3) + \lceil \Delta H \lambda_{\max} \rceil \Delta T_{\text{fail}}$. Thus, for the proposed LLF scheduler, a sufficient time horizon for correct LLF scheduling of messages is:

$$\Delta H \geq \frac{N(\Delta T_{\max} + 3)}{1 - \lambda_{\max} \Delta T_{\text{fail}}} + \Delta T_{\text{fail}}.$$

Note that the above criterion is only a recommendation for the optimal LLF scheduling of soft real-time messages, the timely transmission of hard real-time messages is ensured by the dynamic TDMA scheme, as described below.

5.2. The dynamic TDMA scheme

In order to guarantee the timely transmission of hard real-time messages even in overload situations, all their occurrences have to be predicted and the respective transmission times have to be scheduled in advance in a calendar. Due to the highly critical nature of hard real-time messages, the scheduled transmission times must include worst-case error handling delays and retransmission times under anticipated fault conditions. A study of the worst-case delay times under different fault conditions is given in Rufino and Verissimo (1995).

5.2.1. *The global TDMA calendar*

The reserved time slots are entered into a calendar, which contains all resource reservations. The calendar is contained in each node of the system. This enables the system components to monitor each other's temporal behavior. If a component claims reserved resources at the wrong time, other components will agree on its failure, and can initiate error recovery actions.

5.2.2. *The impact of inaccurate global time*

The scheduling approach for hard real-time communication requires access to a global time reference with bounded inaccuracy. Once a time slot is reserved, the respective action can be started locally. To guarantee that it does not interfere with another time slot, the time reference of all nodes must be synchronized. The lower the clock accuracy, the larger the minimum gap between two subsequent time slots in the global bus schedule (Fig. 3). In order to provide a global time reference with high accuracy, a clock synchronization mechanism has to be applied, e.g. as described by Gergeleit and Streich (1994).

5.2.3. *Exclusive access right in the reserved time-slot*

Efficient handling of time slot reservations is supported by defining the end of the reserved time slot of a message m as its transmission deadline d_m, and calculating its laxity as required by the underlying LLF scheme. The sender of a hard real-time message enforces its exclusive access rights during the reserved time-slot by dynamically increasing the priority of the message according to its laxity. Due to this scheme, a hard real-time message gains the highest possible priority at its latest transmission time. In order to guarantee that different hard real-time messages are always assigned different priorities in the right order, their deadlines must at least differ by the length Δt_p of the priority slot (Fig. 4).

Note that every message may be delayed by one other message, which may have started before it. ΔT_{block} is defined as the longest possible blocking time due to the non-preemptive transmission of an arbitrary message. Since the transmission of a hard real-time message h (with a reserved time slot beginning at S_h) must start before S_h, it must be ready until $S_h - \Delta T_{\text{block}}$ (called the latest ready time LRT_h), and after LRT_h it must always win the arbitration process against all other messages.

From the previous discussion, following requirements are derived for the correct scheduling of hard real-time messages:

(R1) for each occurrence of a hard real-time message h, an exclusive time-slot is reserved, which ends at its deadline d_h ;
(R2) the length ΔR_h of the reserved time-slot of a hard real-time message h is greater or equal to the worst-case transmission time of the message, including all error handling delays and retransmission times under anticipated fault conditions;
(R3) every hard real-time message h is ready for transmission until its latest ready time $LRT_h = S_h - \Delta T_{\text{block}}$;
(R4) during its critical interval $[LRT_h \cdots LST_h]$ (Fig. 5), a hard real-time message h wins the bus arbitration against every other message in the system.

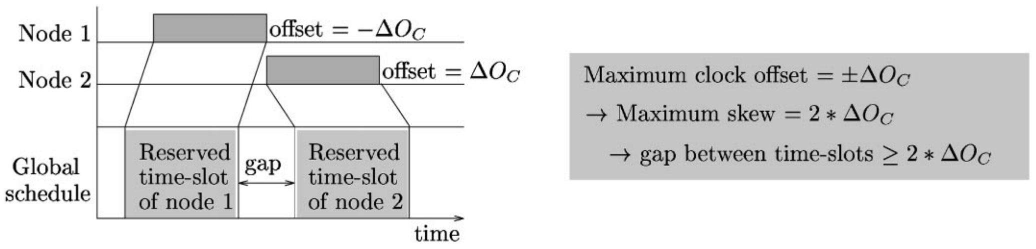
Node 1
Node 2
offset = $-\Delta O_C$
offset = ΔO_C
Global schedule
Reserved time-slot of node 1
gap
Reserved time-slot of node 2
time
Maximum clock offset = $\pm\Delta O_C$
→ Maximum skew = $2 * \Delta O_C$
→ gap between time-slots ≥ $2 * \Delta O_C$

Fig. 3. The minimum gap between reserved time-slots due to clock inaccuracy.

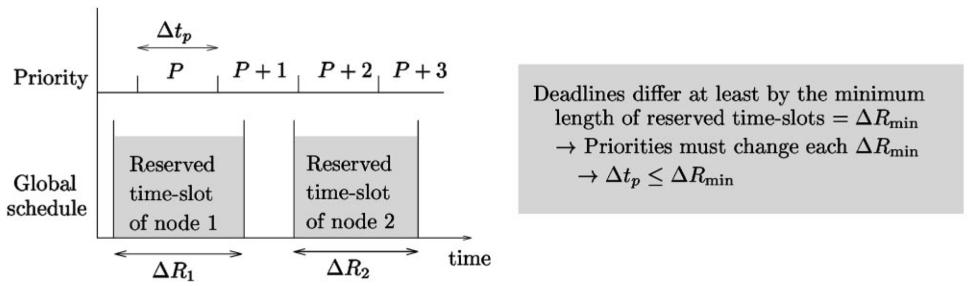
Δt_p
Priority
P
$P+1$
$P+2$
$P+3$
Global schedule
Reserved time-slot of node 1
Reserved time-slot of node 2
ΔR_1
ΔR_2
time
Deadlines differ at least by the minimum length of reserved time-slots = $\Delta R_{\min}$
→ Priorities must change each $\Delta R_{\min}$
→ $\Delta t_p \leq \Delta R_{\min}$

Fig. 4. Maximum length of priority slot Δt_p.

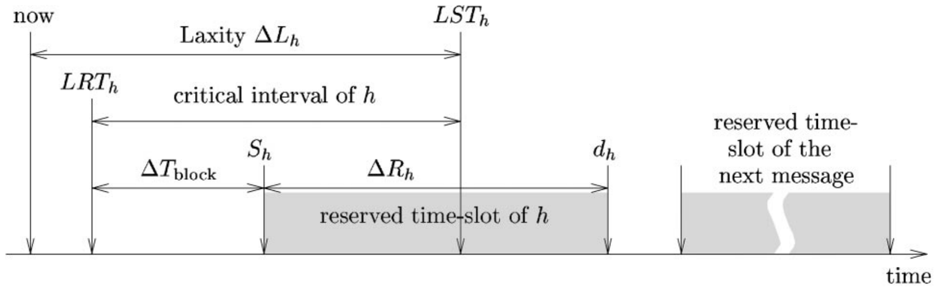
now
LST_h
Laxity ΔL_h
LRT_h
critical interval of h
reserved time-slot of the next message
S_h
d_h
ΔT_{block}
ΔR_h
reserved time-slot of h
time

Fig. 5. The critical interval of a hard real-time message.

Claim 1. If the requirements R1–R4 are fulfilled in a CAN bus, then every hard real-time message will be transmitted timely under anticipated fault conditions.

Proof. Assume that a hard real-time message h occurs with the deadline d_h. Due to R1 and R2 a time-slot $[d_h - \Delta R_h \cdots d_h]$ is reserved for h. Due to R3, h is ready for transmission until $d_h - \Delta R_h - \Delta T_{\text{block}}$. If h is not successfully transmitted until LRT_h, then it participates at least in a bus arbitration at $t \in [LRT_h \cdots S_h]$, and due to R4 h wins the arbitration at t. Due to R2 the time loss caused by successive communication failures is not more than $\Delta R_h - \Delta T_h$. Hence at least one correct message transmission is started in the interval $[t \cdots t + \Delta R_h - \Delta T_h] \subseteq [LRT_h \cdots LST_h]$, and due to R4, h wins the bus for this correct transmission, and finishes until $t + \Delta R_h \leq D_h$. $\square$

The requirements R1, and R2 are ensured by a calendar-based global schedule, which is distributed to all nodes, which also have access to synchronized clocks. In order to reserve the right amount of time for a message in the calendar, the time-slot length must be calculated based on the message length and the fault hypothesis. For a message h with b_h bytes of data, the maximum length of the message including header and bit-stuffing is $L_h = 75 + \lfloor b_h \times 9.6 \rfloor$. Under the assumption of f consecutive transmission failures due to bus/controller errors, the required minimum time-slot length is: $\Delta R_h = (L_h + 18)f + L_h + 3$ bit-times. The requirement, R3 must be satisfied by the synchronization of the local task schedules with the global communication schedule.

To satisfy the requirement R4, the algorithm uses the following rules to ensure that during the critical interval between the latest ready time and the latest transmission start time of a hard real-time message h (Fig. 5), all other messages in the system have lower priorities than h.

(r1) The minimum priority field value for non-real-time messages is higher than the maximum for the real-time messages, i.e. $P^{\text{NRT}}_{\min} > \max\{P^{\text{SRT}}_{\max}, P^{\text{HRT}}_{\max}\}$. Thus, real-time messages always have higher priorities than non-real-time messages.

(r2) The minimum priority field value for soft real-time messages is higher than the maximum of any hard real-time message at the latest ready time, i.e.

$$P^{\text{SRT}}_{\min} > P^{\text{HRT}}_{\min} + \left\lfloor \frac{\Delta R_{\max} + \Delta T_{\text{block}}}{\Delta t_p} \right\rfloor.$$

Thus, after its latest ready time, a hard real-time message has a higher priority than any soft real-time message.

(r3) If the transmission of a hard real-time message is not started until its latest transmission start time, the transmission request is cancelled.

(r4) The priority slot Δt_p *must not be longer* than the minimum distance between the ends of any two consecutive reserved time-slots in the system, minus $\Delta G_{\min}$.

Rule r3 ensures that the transmission of a hard real-time message is either terminated or cancelled until the beginning of the reserved time-slot of the next scheduled hard real-time message, so it does not occupy reserved resources of the next hard real-time message. Rule r4 ensures that — within its critical interval — a hard real-time message has a higher priority than the next scheduled hard real-time message. Thus, within a reserved time-slot, there is no conflict between different hard real-time messages.

5.2.4. *Calculation of the dynamic priorities*

According to rule r1, the total available priority range must be split into two separate sub-ranges for real-time and non-real-time communication. In the current design, the priority field consists of 8 bits, resulting in a range from zero to 255. The priority range $\{192–255\}$ is assigned to the non-real-time communication, i.e. $P^{\text{NRT}}_{\min} = 192$ and $P^{\text{NRT}}_{\max} = 255$.

The priority range $\{0–191\}$ is fully used for the hard real-time communication, i.e. $P^{\text{HRT}}_{\min} = 0$ and $P^{\text{HRT}}_{\max} = 191$. Also, the largest priority value for soft real-time messages is set to $P^{\text{SRT}}_{\max} = 191$. The value $P^{\text{SRT}}_{\min}$ is calculated according to rule r2 as

$$P_{\min}^{\mathrm{SRT}} = P_{\min}^{\mathrm{HRT}} + \left\lfloor \frac{\Delta R_{\max} + \Delta T_{\mathrm{block}}}{\Delta t_p} \right\rfloor + 1$$

$$= \left\lfloor \frac{\Delta R_{\max} + \Delta T_{\mathrm{block}}}{\Delta t_p} \right\rfloor + 1.$$

For example, given a single fault tolerant system, $\Delta R_{\max} \geq 323 \times \Delta bt$ is mandatory. Assuming a $\Delta t_p = 200 \times \Delta bt$, and $\Delta T_{\mathrm{block}} = 154 \times \Delta bt$, the value of $P_{\min}^{\mathrm{SRT}}$ will be 3.

Non-real-time messages are assigned fixed priorities $P_N \in \{P_{\min}^{\mathrm{NRT}} \cdots P_{\max}^{\mathrm{NRT}}\}$. The priority of a hard real-time message with the laxity ΔL is calculated as

$$P_H(\Delta L) = \min \left\{ P_{\min}^{\mathrm{HRT}} + \left\lfloor \frac{\Delta L}{\Delta t_p} \right\rfloor, P_{\min}^{\mathrm{HRT}} \right\}$$

$$= \min \left\{ \left\lfloor \frac{\Delta L}{\Delta t_p} \right\rfloor, 191 \right\}.$$

Soft real-time messages must compete against hard real-time messages according to the LLF scheme, until their priority becomes equal to $P_{\min}^{\mathrm{SRT}}$. After that, the priority of a soft real-time message remains constant. Thus the priority of a soft real-time message with the laxity ΔL is calculated as

$$P_S(\Delta L) = \max \{P_{\min}^{\mathrm{SRT}}, P_H(\Delta L)\}.$$

The assignment of the dynamic priorities is efficiently implemented as follows. The hard and soft real-time messages are inserted into two separate queues, each sorted by deadline. Each time that a new message moves to the head of a queue, its priority is calculated based on its current laxity. Once the priority of a message is assigned, it is updated by periodic decrementation at each Δt_p.

5.3. Schedulability of hard real-time messages

Since the dynamic TDMA scheme guarantees the timely transmission of every hard real-time message by reserving a time-slot, the number of the schedulable hard real-time messages depends on the length of the time-slots and the length of the gaps between time-slots.

In a system tolerating f consecutive transmission failures, the required length of the reserved time-slot of a hard real-time message h can be calculated as

$$\Delta R_h = \Delta T_h + 3 + f\Delta T_{\mathrm{fail}}.$$

As an example, in a single-fault tolerant system, a time-slot of $323 \times \Delta bt$ is required for arbitrary messages. The minimum gap between consecutive time-slots depends on the accuracy of the clock synchronization mechanism in the system. In a CAN bus system, a maximum clock inaccuracy of ± 20 μs can be achieved, e.g. by the algorithm given in Gergeleit and Streich (1994). Assuming a maximum clock inaccuracy of ± 20 μs, the minimum gap $\Delta G_{\min}$ between two consecutive time-slots must be 40 μs. Under these assumptions, a time-slot can be planned each 363 μs. In this case, up to 2754 hard real-time messages per second could be guaranteed.

A sufficient condition for the scheduling of hard real-time messages is that for each message an appropriate time-slot can be inserted into the calendar schedule with the required frequency, with enough of a gap between each pair of consecutive time-slots. This, of course, depends on the application parameters, e.g. individual message periods, etc.

5.4. Computing overhead

The main additional overhead of the hybrid bus scheduling algorithm is a result of the necessary periodic modification of the dynamic priority of the ready-to-send message. Measurements have shown that this task takes about 13 μs when it is implemented as timer interrupt handler on a C167 micro-controller with 20 MHz. Since this is a periodic task that is triggered at each priority slot Δt_p, the processor overhead is *Overhead* $= 13\ \mu\mathrm{s}/\Delta t_p$.

As mentioned in Section 5, the priority slot Δt_p for the hard real-time messages must not be larger than the minimum distance between the ends of any two consecutive reserved time-slots in the system, minus $\Delta G_{\min}$. Using the above parameters, Δt_p may be as large as 323 bit-times. This results in a processing overhead of approximately 4% at the highest bus rate of 1 Mbit/s, and lower overheads at lower bus rates.

6. Conclusion and future research

In order to guarantee timely delivery of hard real-time messages in a CAN network, the authors have introduced a calendar-based scheduling mechanism, which coexists with the EDF scheduling used for soft real-time messages. In contrast to pure EDF scheduling, where resource conflicts may occur due to conflicting deadlines, the presented approach coordinates the deadlines of hard real-time communication by reserving resources in a global calendar. Thus, the timeliness of hard real-time communication is achieved, despite overload failures, by guaranteeing exclusive access rights to the network during the reserved time-slots. This approach guarantees timeliness in presence of communication errors by applying time redundancy. However, in case of non-faulty message transfers, the unused redundant resources are utilized for low-priority communication. This is possible because a hard real-time message releases the network resources after its successful transmission. Thus, optimal resource utilization in fault-free situations as well as in presence of faults is achieved.

The low-level communication protocol presented in this paper will serve as a basis for a high-level atomic multicast protocol with bounded termination time. The concept of the high-level atomic multicast protocol is reported in Livani and Kaiser (1999).

References

Audsley, N.C., Burns, A., Richardson, M.F., & Wellings, A.J. (1992). Deadline monotonic scheduling theory. *Proceedings of the 17th IFAC/IFIP international workshop on real time programming* (pp. 55–60). Bruges.

BOSCH (1991). *CAN Specification Version 2.0*. Robert BOSCH GmbH. Postfach 300240, D-7000 Stuttgart 30.

CiA (1993). CAN application layer (CAL) for industrial applications. *CiA Draft Standards* 201–207.

CiA (1996). CANopen communication profile for industrial systems. *CiA Draft Standard* 301.

Crovella, R.M. (1994). SDS: A CAN protocol for plant floor control. *Proceedings of the first international CAN conference* (pp. 10.2–10.10). Mainz, Germany.

Gergeleit, M., & Streich, H. (1994). Implementing a distributed high-resolution real-time clock using the CAN-bus. *Procedings of the first international CAN conference* (pp. 9.2–9.7). Mainz, Germany.

Kaiser, J., & Livani, M.A. (1998). Invocation of real-time objects in a CAN bus-system. *First international symposium on object-oriented distributed real-time computing systems* (pp. 298–307). Kyoto, Japan.

Kopetz, H., & Grünsteidl, G. (1994). TTP — a time-triggered protocol for fault-tolerant real-time systems. *Computer*, *27*(1), 14–23.

Kopetz, H., & Merker, W. (1985). The architecture of MARS. *Proceedings of the 15th international symposium on fault tolerant computing* (pp. 274–279). Ann Arbor, MI.

Livani, M.A., & Kaiser, J. (1999). A total ordering scheme for real-time multicasts in CAN. *Proceedings of the 24th IFAC/IFIP workshop on real time programming* (pp. 173–178). Dagstuhl, Germany.

Noonen, D., Siegel S., & Maloney, P. (1994). Devicenet application protocol. *Proceedings of the first international CAN conference* (pp. 10.11–10.19). Mainz, Germany.

Rufino, J., & Verissimo, P. (1995). A study on the inaccessibility characteristics of the controller area network. *Proceedings of the second international CAN conference* (pp. 7.12–7.21). London, UK.

Stankovic, J. A., & Ramamritham, K. (1989). The Spring kernel: A new paradigm for real-time operating systems. *ACM Operating Systems Review*, *23*(3), 54–71.

Tindell, K., & Burns, A. (1994). Guaranteeing message latencies on controller area network (CAN). *Proceedings of the first international CAN conference* (pp. 1.2–1.11). Mainz, Germany.

Zuberi, K.M., & Shin, K.G. (1995). Non-preemptive scheduling of messages on controller area network for real-time control applications. *Proceedings of the first IEEE real-time technology and applications symposium* (pp. 240–249).