Universität Ulm

Fakultät für Informatik

Predictable Atomic Multicast in the Controller Area Network (CAN)

Mohammad Ali Livani
Jörg Kaiser
Universität Ulm

Nr. 98-15

Ulmer Informatik-Berichte

Dezember 1998

Predictable Atomic Multicast in the Controller Area Network (CAN)

Mohammad Ali Livani [a] Jörg Kaiser [a]

[a] *University of Ulm, Department of Computer Structures, 89069 Ulm, Germany*

Abstract

The Controller Area Network (CAN) is a broadcast medium providing advanced features, which make it suitable for many real-time applications. Common application layer protocols designed for CAN (e.g. CAL, SDS, DeviceNet) exploit these features in order to provide reliable real-time communication. However, they do not provide a consistent global message ordering in certain fault situations, nor do they consider temporal properties of the application.

This paper presents a multicast protocol which supports timely delivery of messages and guarantees atomic order, under anticipated fault conditions. In order to support causal ordering of events, the paper provides a discussion on establishing Lamport's precedence relation between events by appropriate usage of real-time multicasts.

In a CAN-based distributed system, the restricted communication bandwidth constitutes a serious bottle-neck. Therefore an important feature of this multicast protocol is to achieve optimal protocol termination time while requiring minimum communication overhead.

Key words: Real-time communication; fault-tolerance; multicasting; total order; CAN

1 Introduction

Future computer systems will, to a large extent, monitor and control real-world processes. This results in an inevitable demand for timeliness and reliability. Distributed systems which inherently provide extensibility and immunity against single failures, are an adequate architecture to cope with spatially distributed real world applications. Moreover, the availability of inexpensive, powerful micro-controllers promotes distributed solutions.

Group communication is the basic mechanism to coordinate distributed activities. In order to control the competition and cooperation among distributed processes, and to allow for object replication, atomic delivery of multicast messages is inevitable. In this paper, the following definition of atomic multicast delivery is used.

Definition 1 Atomic multicast delivery – *Let G be a multicast group, i.e. a set of objects which must receive a multicast message. Let $del_i(m)$ denote the delivery of a message m to an object i and '$\longrightarrow$' define the precedence relation. Then two messages m and m' are delivered atomically to the group G, if and only if the following holds:*

$$\forall i,j:\ i,j \in G \ \wedge\ i,j \text{ non-faulty} \\ \Rightarrow (\ del_i(m) \longrightarrow del_i(m') \Leftrightarrow (del_j(m) \longrightarrow del_j(m'))$$

In other words, atomic delivery of multicast messages implies two properties:

- *Consistent Delivery*: if a multicast message is received by a non-faulty group member, then it is received by all non-faulty group members.
- *Consistent Ordering*: messages received by different non-faulty group members, are received in the same order.

Achieving the consistent delivery requires either a two-phased commit protocol with a considerable acknowledgment overhead (Babaoglu and Drummond, 1985; Birman and Joseph, 1987; Chang and Maxemchuk, 1984), or a mechanism to retransmit each message so many times that the probability of losing all copies is negligible (Cristian, 1990; Livani, 1998; Rufino *et al.*, 1998).

Consensus on the ordering of the messages comes free with the two-phased commit protocol. But if the multiple transmission approach is applied, an additional mechanism for establishing a unique order of the messages must be used. (Schneider, 1990) has described an approach to achieve consensus on the message ordering among a group of receivers, using a time-dependent unique identifier of messages. An approach which exploits knowledge about the delivery deadlines of the messages, has been introduced by (Cristian *et al.*, 1985). A similar mechanism has been proposed for embedded applications by (Zuberi and Shin, 1996). These approaches, however, rely on the timely and reliable delivery of all messages.

The ordering scheme presented in this paper considers also the late transmission of soft real-time messages, which is a timing failure of the message transmission, but does not lead to a system failure. As shown in section 3, it is possible to schedule the real-time communication in a CAN-bus system so that hard real-time messages are always transmitted timely while soft trans-

mission deadlines are missed in overload situations. In such a system it is not possible to guarantee a total delivery order among all hard and soft real-time messages.

To see the reason of this restriction consider a hard real-time message H, a soft real-time message S, and two nodes N_1 and N_2. Since no latest delivery time can be guaranteed for S, its consistent delivery must depend on the reception of a signal σ from the bus (e.g. S itself, a commit packet, or another packet depending on the protocol). Assume that N_1 receives σ at time t_1, and N_2 does not because of a single communication error. If N_1 and N_2 receive H after t_1 and N_2 does not receive σ until d_H (i.e. the deadline of H) due to bus overload, then at d_H, N_1 must deliver S before H, and N_2 must deliver H before S.

Due to the restriction mentioned above, this paper proposes an ordering algorithm, which establishes a total order for the messages of each class (i.e. hard and soft real-time) separately.

The paper is organized as follows: section 2 introduces some properties of the CAN bus which are essential to understand the approach. Section 3 presents the deadline-based timely message delivery scheme. Section 4 introduces the deadline-based total ordering scheme. Section 5 discusses how to reflect Lamport's precedence relation using the deadline-based total order. A summary concludes the paper.

2 Some Properties of CAN

The CAN-bus (BOSCH, 1991) is a priority bus targeted to operate in a noisy environment with speeds of up to 1 Mbit/s, exchanging small real-time control messages.

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-bit or 29-bit identifier (depending on the "standard" or "extended" format - Figure 1). If at this time a node sends a '1' and senses

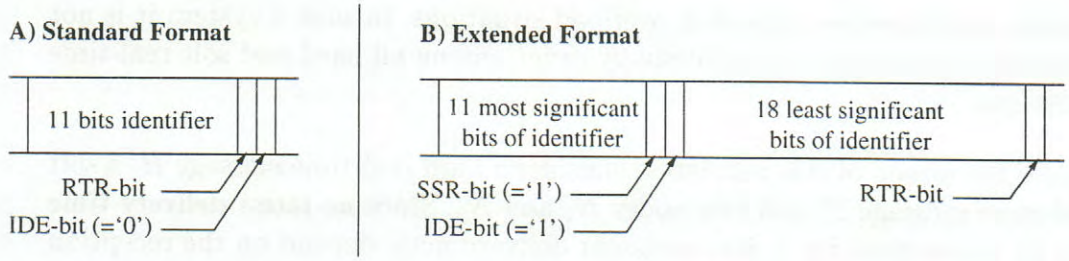


Fig. 1. The CAN arbitration field

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.

The CAN protocol provides efficient hardware-implemented error handling, which is based on various error detection mechanisms with a very high total coverage, and an approach to immediately signaling the error condition. These features of the CAN protocol ensure atomic broadcast delivery in most fault situations. However, if an error occurs within the transmission time of the two last bits of a data frame, some receivers might accept and other receivers reject the frame. Although the sender (or in case of immediate sender crash an alternative mechanism like Eager Diffusion (Rufino *et al.*, 1998) or Shadow Retransmitter (Livani, 1998)) will retransmit the frame, the consistent ordering of the incoming messages in different sites is not trivial due to the possible transmission of other (high-priority) messages between the first and the second transmission of a frame. Another problem caused by such errors is the duplicate frame reception by some receivers.

In such situations, commercially available application level protocols for CAN, like CAL (CiA, 1993), SDS (Crovella, 1994), and DeviceNet (Noonen *et al.*, 1994) manage to discard frame duplicates, but they fail to provide consistent message ordering among a group of receivers.

3 The Deadline-based Timely Message Transmission Mechanism

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

8 bits	8 bits	13 bits
Priority	TxNode	Message name

Fig. 2. Partitioning of a CAN-message identifier

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 and Merker, 1985; Kopetz and 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.

3.1 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. Figure 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's 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.

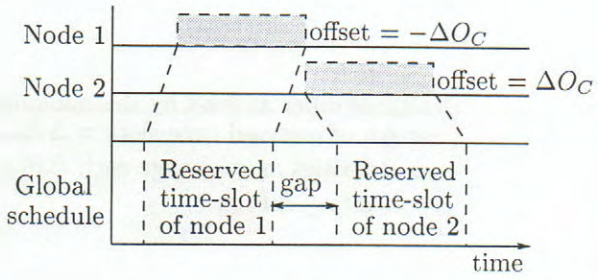
The period Δt_p is called the priority slot. Since all laxity values $\Delta L \geq (P_{\max} - P_{\min}) * \Delta t_p$ are mapped to the same priority, the priority-based dispatcher (i.e. the CAN arbitration mechanism) cannot distinguish different laxity values which are greater or equal $(P_{\max} - P_{\min}) * \Delta t_p$. 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.



Maximum clock offset = $\pm\Delta O_C$

$\rightarrow$ Maximum skew = $2 * \Delta O_C$

$\rightarrow$ gap between time-slots $\geq 2 * \Delta O_C$

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

3.2 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).

3.2.1 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.

3.2.2 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 (Figure 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).

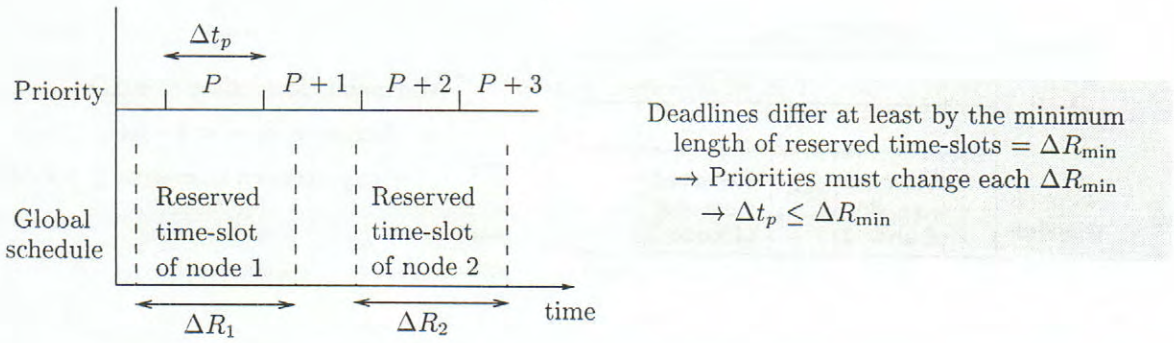


Fig. 4. Maximum length of priority slot Δt_p

3.2.3 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 (Figure 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 occurence 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 $[LRT_h \cdots LST_h]$ (Figure 5), a hard real-time

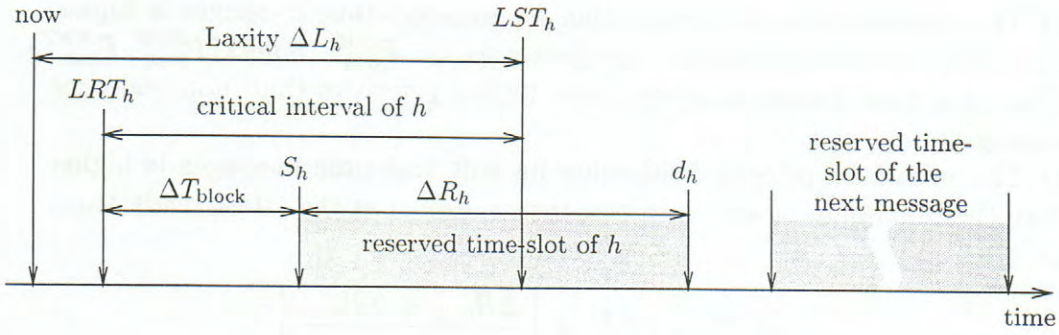


Fig. 5. The critical interval of a hard real-time message

message h wins the bus arbitration against every other message in the system.

Claim 2 *If the requirements R1 through 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, m wins the bus for this correct transmission, and finishes until $t + \Delta R_h \leq D_m$. □

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 $L_h = 75 + \lfloor b_h * 9.6 \rfloor$. 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 (Figure 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. $P_{\min}^{NRT} > \max\{P_{\max}^{SRT}, P_{\max}^{HRT}\}$. 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.

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

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 $\Delta G_{\min}$.

The 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. The 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.

3.2.4 Calculation of the dynamic priorities.

According to the 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 $\{192 \cdots 255\}$ is assigned to the non real-time communication, i.e. $P_{\min}^{NRT} = 192$ and $P_{\max}^{NRT} = 255$.

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

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

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

will be 3.

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

$$P_H(\Delta L) = min\left\{P_{\min}^{HRT} + \left\lfloor \frac{\Delta L}{\Delta t_p} \right\rfloor, P_{\max}^{HRT}\right\} = min\left\{\left\lfloor \frac{\Delta L}{\Delta t_p} \right\rfloor, 191\right\}$$

Soft real-time messages must compete against hard real-time messages according to the LLF scheme, until their priority becomes equal to $P_{\min}^{SRT}$. 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:

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

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.

3.3 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:

$$\Delta R_h = \Delta T_h + 3 + f * \Delta T_{\text{fail}}$$

As an example, in a single-fault tolerant system, a time-slot of $323 * \Delta bt$ 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 $\pm 20\mu s$ can be achieved, e.g. by the algorithm given in (Gergeleit and Streich, 1994). Assuming a maximum clock inaccuracy of $\pm 20\mu s$, the minimum gap $\Delta G_{\min}$ between two consecutive time-slots must be $40\mu s$. Under these assumptions, a time-slot can be planned each $363\mu s$. 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.

4 The Deadline-Based Total Ordering Scheme

Based on a feasible and flexible real-time scheduling policy as described in the previous section, combined with a reliable broadcast delivery mechanism (Livani, 1998; Rufino *et al.*, 1998), following assumptions can be made about the communication system in a CAN bus:

(A1) Hard real-time messages are transmitted timely (i.e. until their transmission deadline) under anticipated fault scenarios, even in overload situations.

(A2) Soft real-time messages are scheduled by static or dynamic priorities, but their deadlines may be missed in overload situations.

(A3) If a non-faulty receiver receives a message, then eventually all non-faulty receivers receive the message.

This section introduces a scheme to achieve globally consistent ordering of multicast messages in a CAN-based system. The algorithm presented here relies on the assumptions A1 through A3 in order to achieve atomic multicast with minimum communication overhead, based on the knowledge about the message transmission deadlines. The transmission deadline of a message denotes the time, at which the message must be successfully transmitted to all non-faulty destination sites. Since the transmission deadline is tightly related to the time, where the sender expects the message delivery to all receiving application objects, this ordering mechanism allows an application specific ordering, which is related to temporal requirements. Although this approach is similar to some other ones known from much previous work (Cristian *et al.*, 1985; Schneider, 1990), it considers the late transmission of soft real-time messages in overload situations. Another advantage of this scheme is that it needs no additional communication to establish a globally consistent ordering decision.

Let m and m' be two real-time messages with transmission deadlines d_m and $d_{m'}$. Then the deadline-based ordering algorithm implements the following rule:

(O1) $d_m < d_{m'} \Rightarrow del_j(m) \longrightarrow del_j(m')$

This means that if the deadline of the message m is before the deadline of the message m', then m must be delivered to destination objects before m'. If the underlying protocol layer adjusts the transmission deadlines of hard real-time messages to reserved time-slots (cf. section 3), then the delivery order of hard real-time messages can be always established by this rule.

However, when the deadlines of different messages are equal, the following rule must be used to ensure the globally consistent decision on delivery order of messages.

(O2) $d_m = d_{m'} \Rightarrow (del_j(m) \longrightarrow del_j(m') \Leftrightarrow header_m < header_{m'})$

Where $header_m$ consists of a tuple (d_m, $sender_m$, $subject_m$) and identifies a message uniquely. This rule requires that different messages sent by the same sender with the same subject have different transmission deadlines, otherwise an additional sequence number (or "toggle-bit") must be provided in the header. Note that this requirement is also necessary for distinguishing duplicate frames from successive frames.

4.1 Protocol Termination and End-to-End Delivery

An atomic multicast protocol may terminate and deliver a message to destination objects, if and only if a) the message is received and accepted by all non-faulty destination sites, and b) the message can be consistently ordered, which means that no other message will arrive later, which must precede the current message according to the ordering criteria.

a) Consistent Message Reception in CAN

In order to minimize the communication overhead, the nodes must not exchange explicit information about the status of their message queues. Thus they have to conclude this information from implicit knowledge. Since the underlying message transfer protocol guarantees the timely transmission of hard real-time messages under anticipated fault conditions, the protocol can assume that a hard real-time message is consistently transmitted to all receivers at its deadline. Hence a receiver can deliver the message to the destination objects at the transmission deadline.

For soft real-time messages, however, the transmission deadline may be missed because of bus overload, and hence, late transmission of the message is still possible. In these situations, the deadline cannot be used to specify the time when the message may be delivered. However, in such situations the following considerations lead to a decision criterion:

Claim 3 *If a node has received a message m, and then another message with lower priority is observed on the CAN bus, or the bus is idle, then the sender of m will not retransmit it in future.*

PROOF. Assume that a message m is transmitted at least once on the CAN bus. Further, assume that the sending CAN controller still attempts to retransmit m due to the inconsistent transmission. According to the CAN specification (BOSCH, 1991), the sender will try to retransmit m "immediately", thus no bus idle period will be observed before the retransmission of m. Furthermore, no lower-priority message will be transmitted before the retransmission of m, because m will win the arbitration process against any lower-priority message. □

As a consequence, the receivers of a soft real-time message m can be sure that either the sender is crashed, or the transmission was successful, as soon as they detect either a bus idle period or a lower-priority message after receiving m.

If a frame is accepted by a subset of the receivers, and rejected by other receivers, the immediate sender crash may lead to inconsistent message reception. This failure must be tolerated by having some nodes retransmit the frame. The Eager Diffusion Protocol (Rufino *et al.*, 1998) retransmits every frame at least by one receiver. However, due to processing delay the Eager Diffusion Protocol cannot guarantee the immediate retransmission. The ordering scheme presented in this paper relies on dedicated Shadow Retransmitters (Livani, 1998), which guarantee immediate retransmission whenever an inconsistent message reception is possible. The Shadow retransmitters also transmit an ultra-low priority frame (called trailer) at the end of each non-broken sequence of CAN frames. So an idle bus can be detected by observing a trailer on the bus.

From the previous discussion, following properties are derived:

(P1) Any receiver of a hard real-time message m with deadline d_m, can assume that m will be received by all sites until d_m.

(P2) Any receiver of a soft real-time message m can assume that the message has been received by all sites, if it observes a frame with a lower priority on the bus after receiving m.

b) Achieving a Consistent Message Order in CAN

In case of hard real-time messages, the knowledge of time is sufficient for stabilizing the ordering decision: after a deadline d_m, no hard real-time message with an earlier deadline will arrive. But this statement is not true in case of soft

real-time messages. Here, the following assumption constitutes the constraint necessary to find a stable ordering decision criterion:

(A4) Every real-time message m is ready to transmit before $d_m - \Delta T_{\max}$, with $\Delta T_{\max}$ being the maximum time required for a single transmission of an arbitrary frame.

The assumption (A4) concludes the following:

Claim 4 *Assume a deadline-based message priority scheme (e.g. as described in section 3), where a soft real-time message with a higher priority has an earlier deadline than a soft real-time message with a lower priority. If after the transmission deadline d_m of a soft real-time message m another message with lower priority is transmitted on the CAN bus, or the bus is idle, then no other soft real-time message m' with a deadline $d_{m'} \leq d_m$, will be transmitted later.*

PROOF. Assume that a message m' with deadline $d_{m'} \leq d_m$ is pending for transmission at the time $t > d_m$. Because of the deadline-based priority assignment the priority of m' is not lower than the priority of m. Due to (A4), m' has been pending for transmission at least since $d_{m'} - \Delta T_{\max}$, hence at least since $d_m - \Delta T_{\max}$. Hence the sender of m' must have been trying to transmit m' at the beginning of every bus-idle period since $d_m - \Delta T_{\max}$. Therefore no idle bus can be observed between d_m and the successful transmission of m'. Also, no lower-priority message can be transmitted on the bus before m', because m' wins the arbitration process against any lower-priority message. Thus, if after d_m a message with a lower priority than m is transmitted on the CAN bus, or the bus is idle, then no message m' with $d_{m'} \leq d_m$ will be transmitted later. □

From the previous discussion, following property is derived:

(P3) For any soft real-time message m, no preceding soft real-time message will arrive later, if after the transmission deadline d_m a lower-priority message is observed on the bus.

4.2 Atomic Multicast Delivery Algorithm

The proposed atomic multicast delivery algorithm is based on the following rules:

Order: Messages are ordered by their deadlines. In case of equal deadlines, the value of the message header is used for the order decision.

Time1: every received hard real-time message is delivered at its deadline. (This realizes the deadline-based order implicitly)

Time2: Every received soft real-time message must be delivered as soon as either another message with later deadline, or a bus idle time is observed after its transmission deadline.

Due to the rule (Time2), the delivery of a soft real-time message m may be delayed until $d_m + \Delta T_{\max}$. This must be considered when calculating the transmission deadline of any soft real-time message.

In the following, the atomic multicast delivery algorithm is presented. The algorithm assumes the availability of a real-time clock, which is synchronized globally with a bounded inaccuracy.

```
Atomic_multicast_delivery
initialization:
  SRT_q ← empty_q;
  HRT_q ← empty_q;
  next_HRT_delivery ← eternity;
receive(m,t): /* m is sent or received at t */
  if m.dest_group ∈ my_groups and m.category = HRT then
    HRT_q.insert (m);
    if m.dl < next_HRT_delivery then
      next_HRT_delivery ← m.dl;
      set_wakeup (next_HRT_delivery);
  if m.dest_group ∈ my_groups and m.category = SRT then
    SRT_q.insert (m);
    if m.dl ≤ t then
    /* overload! Deliver preceding SRTM */
      while SRT_q.head.dl < m.dl or SRT_q.head < m do
        mx ← SRT_q.gethead;
        SRT_deliver (mx, mx.dest_group);
    else
    /* Deliver SRTM with passed DL */
      while SRT_q.head.dl < t do
        mx ← SRT_q.gethead;
        SRT_deliver (mx, mx.dest_group);
end receive;
wakeup(): /* invoked when the wake–up
             time is reached */
  mx ← HRT_q.gethead;
  HRT_deliver (mx, mx.dest_group);
  if HRT_q ≠ empty_q then
```

```
    next_HRT_delivery ← HRT_q.head.dl;
    set_wakeup (next_HRT_delivery);
  else
    next_HRT_delivery ← eternity;
end wakeup;
bus_idle(t): /* invoked if at time t a
                trailer frame is received */
  while SRT_q.head.dl < t do
  /* Deliver all SRTM with passed DL */
    mx ← SRT_q.gethead;
    SRT_deliver (mx, mx.dest_group);
end bus_idle;
end Atomic_multicast_delivery
```

5 Global Ordering of Events

The deadline-based ordering of multicast messages enables a consistent global order between different events observed in the distributed system. In this paper an event is called a global event, if the object which observes it, sends an atomic multicast to enforce a global observation of the event by all concerning objects. As seen by application objects, the order of global events is the same as the globally consistent delivery order of their notification messages. This globally consistent delivery order of messages is established by the atomic multicast protocol described in section 3.2. Note that if a set of events has to be globally ordered, then all multicast messages propagating those events must belong to the same class (hard or soft real-time).

5.1 Achieving Causal Ordering of Events

If global events have to be ordered according to their causality, following cases can be observed:

A: Let e and e' be two global events locally observed by an object i, with a causal precedence relation $e \longrightarrow e'$, and let m and m' be messages, which propagate the events e and e' in the system. Let d_m denote the deadline of m. In order to globally agree on the order $e \longrightarrow e'$, for every object j, the relation $del_j(m) \longrightarrow del_j(m')$ is sufficient. Hence, due to the rule (O1), the relation $d_m < d_{m'}$ is sufficient. This means, that in order to reflect a precedence order $e \longrightarrow e'$ globally, i must assign deadlines $d_m < d_{m'}$ to the messages propagating e and e' in the system.

B: Let e and e' be two events observed by two different objects i and j, and

$e \longrightarrow e'$, then (according to Lamport's definition of precedence relation (Lamport, 1978), and the discussion in section 3.1) following conditions hold:

(1) Event e is observed by i at local time $T^i(e)$.

(2) Event e' is observed by j at local time $T^j(e')$.

(3) There is a message m sent by i at the local time $T^i(send_i(m))$, which propagates the event e, where $T^i(e) \leq T^i(send_i(m)) < d_m$.

(4) There is a message m' sent by j at the local time $T^j(send_j(m'))$, which propagates the event e', where $T^j(e') \leq T^j(send_j(m')) < d_{m'}$.

(5) If j receives m, then it receives m at the local time $T^j(del_j(m))$, where $d_m \leq T^j(del_j(m))$, and $T^j(del_j(m)) \leq T^j(e')$. Thus due to (4) it follows that $d_m < d_{m'}$.

(6) If j does not receive m, then there is a set of messages $m_1, m_2, m_3, \cdots, m_n$, and a set of objects $k_1, k_2, k_3, \cdots, k_n$, where firstly, k_1 receives m and sends m_1, k_2 receives m_1 and sends $m_2, \cdots, k_n$ receives m_{n-1} and sends m_n, and j receives m_n. Secondly, $del_{k_1}(m) \longrightarrow send_{k_1}(m_1)$, $del_{k_2}(m_1) \longrightarrow send_{k_2}(m_2)$, $\cdots$, and $del_{k_n}(m_{n-1}) \longrightarrow send_{k_n}(m_n)$.

Thirdly, $T^j(del_j(m_n)) \leq T^j(e')$. In this case, $d_m \leq T^{k_1}(del_{k_1}(m)) < T^{k_1}(send_{k_1}(m_1)) < d_{m_1} \leq T^{k_2}(del_{k_2}(m_1)) < T^{k_2}(send_{k_2}(m_2)) < d_{m_2} \cdots < d_{m_n} \leq T^j(del_j(m_n)) \leq T^j(e')$. Again, due to (4) it follows that $d_m < d_{m'}$.

In either case (i.e. A; B–5; B–6), every object l in the system which receives m and m', will establish the delivery order $del_l(m) \longrightarrow del_l(m')$ due to the relation $d_m < d_{m'}$. Consequently, l will establish the precedence order $e \longrightarrow e'$. Thus, if e and e' are two global events observed in the distributed system, and $e \longrightarrow e'$ according to Lamport's definition of precedence relation, then all non-faulty objects in the system – which are addressed by messages propagating e and e' – will be notified about e before e', and hence they will establish consistently the precedence order $e \longrightarrow e'$.

6 Conclusion

The paper introduced a real-time multicast ordering scheme, which achieves consistent delivery ordering of multicast messages using the message transmission deadlines.

The protocol termination time depends on the message class. The protocol relies on a medium access protocol, which guarantees timely and reliable transmission of hard real-time messages to all destination nodes (Livani, 1998). Hence, for a hard real-time message the protocol terminates at the transmission deadline and delivers the message to destination objects timely. Under 'normal' load conditions, the protocol delivers soft real-time messages up to

one frame transmission time later than their transmission deadlines (cf. rule Time2 in section 3.2). However, in overload situations, the protocol may delay the delivery of soft real-time messages until the end of the overload period plus one frame transmission time.

The mechanism ensures consistent multicast ordering in presence of communication failures leading to inconsistent global view of the CAN bus status among non-faulty sites. Common high-level communication protocols designed for CAN, do not provide consistent multicast delivery order among all application objects in these failure situations.

It was shown, that causal order between events can be globally established by using the deadline-based ordering approach.

An important benefit of this scheme is its efficiency: the algorithm requires no additional communication, like acknowledgements etc. In the Controller Area Network, communication bandwidth is a serious bottle-neck. While the computing power of hardware nodes is increasing rapidly, the bandwidth of CAN will remain limited to a maximum of 1 Mbits/sec. This was the main motivation to develop a multicast scheme which minimizes communication amount for the price of some computational overhead.

References

Babaoglu, Ö. and R. Drummond (1985). Streets of byzantium: Network architectures for fast reliable broadcasts. *IEEE Tr. Software Eng.* **11**(6), 546–554.

Birman, K.P. and T.A. Joseph (1987). Reliable communication in the presence of failures. *ACM Tr. on Computer Systems* **5**(1), 47–76.

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

Chang, J.M. and N.F. Maxemchuk (1984). Reliable broadcast protocols. *ACM Tr. on Computer Systems* **2**(3), 251–273.

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

Cristian, F. (1990). Synchronous atomic broadcast for redundant broadcast channels. *The Journal of Real-Time Systems* **2**, 195–212.

Cristian, F., H. Aghili, R. Strong and D. Dolev (1985). Atomic broadcast: From simple message diffusion to byzantine agreement. In: *Proc. of the 15th Int. Symposium on Fault Tolerant Computing.* Ann Arbor, MI. pp. 200–206.

Crovella, R.M. (1994). SDS: A CAN protocol for plant floor control. In: *1st International CAN Conference.* Mainz, Germany. pp. 10.2–10.10.

Gergeleit, M. and H. Streich (1994). Implementing a distributed high-resolution real-time clock using the CAN-bus. In: *1st International CAN Conference.* Mainz, Germany. pp. 9.2–9.7.

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

Kopetz, H. and W. Merker (1985). The architecture of MARS. In: *Proc. of the 15th Int. Symposium on Fault Tolerant Computing.* Ann Arbor, MI. pp. 274–279.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of ACM* **21**(7), 558–565.

Livani, M.A. (1998). SHARE: A transparent mechanism for reliable broadcast delivery in CAN. *Informatik Bericht 98-14, University of Ulm.*

Noonen, D., S. Siegel and P. Maloney (1994). Devicenet application protocol. In: *1st International CAN Conference.* Mainz, Germany. pp. 10.11–10.19.

Rufino, J. and P. Verissimo (1995). A study on the inaccessibility characteristics of the controller area network. In: *Proc. 2nd International CAN Conference.* London, UK. pp. 7.12–7.21.

Rufino, J., P. Verissimo, G. Arroz, C. Almeida and L. Rodrigues (1998). Fault-tolerant broadcasts in CAN. In: *Proc. 28th Int. Symposium on Fault Tolerant Computing.* Munich, Germany. pp. 150–159.

Schneider, F. (1990). Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys* **22**(4), 299–319.

Zuberi, K.M. and K.G. Shin (1996). A causal message ordering scheme for distributed embedded real-time systems. In: *Proc. Symp. on Reliable and Distributed Systems.* Niagare-on-the-Lake, Canada. pp. 210–219.

Liste der bisher erschienenen Ulmer Informatik-Berichte

Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm

Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

91-02* *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* *Alfons Geser*
Relative Termination

91-04* *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

91-06* *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

91-07* *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

91-08* *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

92-01* *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

92-05* *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

92-06* *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

92-07* *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

92-08* *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

92-09* *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

92-14* *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Ganer*
Rechnerunterstützung fr die konzeptuelle Modellierung

93-07 *Ullrich Keler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Khnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman, Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien
für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)