A Symmetric MAC Protocol for CSMA Busses in Dynamic Distributed Real-time Systems

Mohammad Ali Livani, Jörg Kaiser
University of Ulm, Department of Computer Structures, 89069 Ulm, Germany
{Mohammad, Kaiser}@informatik.uni-ulm.de

Abstract

By using a hybrid scheduling algorithm consisting of static offline scheduling and dynamic online scheduling, hard deadlines can be guaranteed, while achieving optimal resource utilization by soft real-time activities. This paper introduces a medium access control (MAC) protocol for a CSMA bus, which supports the hybrid scheduling of hard and soft real-time messages on the bus. The key issues considered here, are distinguishing hard and soft real-time constraints, achieving high resource utilization, and avoiding single points of failure by a symmetric distributed medium access control scheme.

1. Introduction

The application domain of real-time computing systems is growing into areas, where cost-effectiveness is a mandatory requirement. Typical examples are drive-by-wire, vehicle body electronics, intelligent home, and multimedia. The demand for cost-effective real-time solutions and the availability of inexpensive microprocessors with integrated network interface, have led to the high popularity of distributed real-time systems, which are based on field busses and LAN's with a real-time communication protocol.

In contrast to non real-time systems that are optimized for high throughput, real-time systems must primarily exhibit temporal predictability. This means that real-time computing results must be provided within a specified response time. Temporal specification of computing activities can be roughly categorized as follows: hard real-time activities must always meet their (hard) deadlines, otherwise it comes to a serious or even fatal system failure. Soft real-time activities should meet their (soft) deadlines whenever possible, but they may be completed later, too. Non real-time activities do not have a deadline, and must proceed whenever no real-time activities need the computing resources.

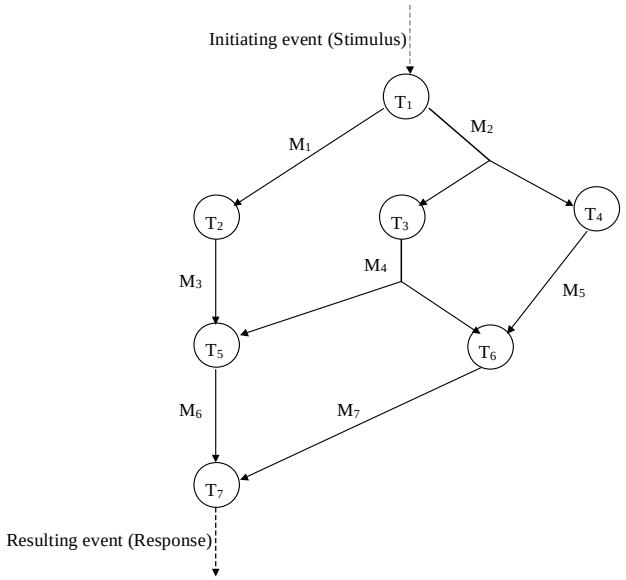

Figure 1. A real-time transaction consisting of tasks and messages

Dynamic distributed real-time systems are characterized by a mixture of periodic, sporadic, and aperiodic real-time transactions, which may have hard or soft deadlines. A real-time transaction consists of several tasks with precedence relations among them. Data and control flow between tasks is usually realized by message exchange (cf. Figure 1). Tasks and messages inherit the period and deadline characteristics of the respective real-time transactions.

In order to guarantee hard deadlines, the minimum invocation period and the maximum execution times of hard real-time activities must be known. This implies that only periodic and sporadic activities may have hard deadlines. There are mature research results, which enable the calculation of the maximum execution time of tasks [13],[14], and the generation of a feasible offline schedule [3], assuming a TDMA protocol on the system

bus. By using these methods, adequate resources are reserved to meet hard deadlines even in worst-case anticipated fault and load scenarios.

Since resource reservation is based on pessimistic assumptions, reserved resources are often unused. Moreover, there are usually some non-reserved resources remaining. In a strictly time-triggered system like MARS [4],[1], it is not possible to exploit system resources, without reserving them by the offline scheduler. In contrast, a dynamic system, e.g. MARUTI [8], can achieve optimal resource utilization by Earliest Deadline First (EDF) scheduling mechanism.

In a complex real-time application, like drive-by-wire, different distributed computations with hard and soft deadlines co-exist. Periodic hard real-time activities like motor management and by-wire steering must be performed periodically with minimal jitter, and critical sporadic activities like anti-lock brake control or electronic stabilization program have to be guaranteed with their highest occurrence rates. But activities like gear change are less critical, and may be delayed under certain circumstances, e.g. when electronic stabilization program is activated.

Thus there are distributed computations, which must be performed by best effort 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 with their highest anticipated frequency, then some reserved resources remain idle, and can be used for soft real-time computation. If less errors occur than anticipated by the system's design, then probably some redundantly reserved resources for hard real-time activities remain unused, and may also be exploited by 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. Such a scheduling mechanism must be supported by an appropriate MAC protocol, which implements the system scheduling policy on the bus resource. such a MAC protocol must guarantee the timely transmission of hard real-time messages by offline resource reservation, while utilizing the bus resource optimally for the best-effort transmission of soft real-time messages, e.g. by EDF. An example of coexistence of offline and online bus scheduling in the Controller Area Network, has been presented in [10].

This paper introduces a scheme to merge both static (calendar-based) and dynamic (deadline-based) scheduling mechanisms into a common priority scheme, which enables common scheduling of hard and soft real-time messages by a priority-based bus arbitration mechanism.

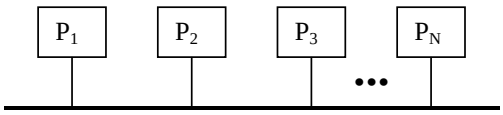

Figure 2. A distributed system based on a bus

While a priority-based dispatcher is included in almost every real-time executive, a priority-based access regulation in a CSMA bus must be achieved with additional effort. For this reason, we propose a derivative of the Virtual Time CSMA [12],[16], which enables common dispatching of hard and soft real-time messages on a CSMA network. The key issues considered in this paper are 1) distinguishing hard real-time, soft real-time, and non real-time constraints, 2) achieving high resource utilization, and 3) enabling fault-tolerance by avoiding single points of failure in the distributed medium access scheme.

The paper is organized as follows: section 2 introduces the system model. Section 3 provides a brief description of the Virtual Time (VT) CSMA. In section 4 we describe a hybrid approach for real-time scheduling of messages with different criticality classes in Ethernet, using the VTCSMA approach. In section 5 some simulation results are presented. A summary concludes the paper.

2. The system model

We consider a system hardware consisting of a set of processing nodes, all attached to a CSMA bus (e.g. Ethernet), as illustrated in Figure 2. The system is heterogeneous, i.e. the nodes may have different architectures, from low-cost 8-bit micro-controllers to multiprocessor workstations. Low-cost micro-controllers are attached to sensors and actuators, building up smart sensors and actuators with well-defined and vendor-independent functional interface, and capable to communicate over the system bus. By attaching sensor-actuator combinations to micro-controllers, elementary control loops can be realized as a local functionality, and only monitoring and control information of higher levels have to be communicated on the bus. This reduces the expensive cabling and simplifies the applications system's design.

As already mentioned in section 1, a real-time transaction (RT) consists of several tasks with precedence relations, exchanging data and control by messages. The starting task of a RT is triggered by at least one event, e.g. an external signal or a clock interrupt.

Scheduling distributed real-time activities requires local scheduling of tasks at each node as well as global scheduling of messages on the bus. In order to guarantee the timely completion of a hard real-time RT, all its associated messages must be scheduled as hard real-time. Similarly, messages associated with a soft real-time RT, must be scheduled with soft deadlines. Treating such messages as hard real-time is unnecessary, and may lead to bandwidth shortage.

In a completely static system, a global calendar is available and each node has its relevant entries referring to its message transmission times in a global time scale. A message may only be transmitted according to this schedule [4],[1],[6]. In a more dynamic system where hard real-time, soft real-time, and non-real-time messages coexist, things are more complicated. If the bus is free, a soft real-time message may be sent. In this case, it must be guaranteed that it does not cause a timing failure of a hard real-time message.

3. The VTCSMA protocols

The basic idea of the VTCSMA scheme [12] is that different priorities can be realized in a CSMA bus by different contention periods. The higher the priority of a message, the shorter its contention period must be. The VTCSMA algorithm is a general approach, which can map different message parameters (e.g. deadline or arrival time) onto a contention period, hence prioritizing messages and scheduling them according to different strategies (e.g. earliest-deadline-first or first-come-first-serve) [16]. In following, we briefly describe the deadline-based variant of the algorithm, which is called VTCSMA-D.

3.1. The VTCSMA-D protocol

Each time the bus becomes free, the virtual start time of messages is determined by taking their deadlines. The current virtual time is set equal to the current real time, and the virtual clock is started with a speed higher than the real time (i.e. at a rate $\eta>1$). If the bus remains free until the virtual time reaches the virtual start time of a message, then the message is started. If the bus becomes busy before a node starts sending a message, the virtual clock is stopped. If a collision occurs while a node is trying to send a message, the message is either retransmitted immediately (with a probability P) or the virtual time is promoted to a random value between the current virtual time and the virtual start time of the message. Although non-deterministic, this collision handling mechanism results in efficient utilization of the bus, as shown by Zhao and Ramamritham [16].

When using the deadline-based variant of VTCSMA, even messages with different deadlines may collide, if local clocks of their senders differ by the same amount. Correct scheduling of such messages is guaranteed, if their deadlines differ by more than the maximum clock skew between non-faulty nodes, which can be bounded by an appropriate clock synchronization mechanism [5]. In this paper we apply some modifications to VTCSMA-D, in order to schedule hard and soft real-time messages on a CSMA bus.

4. Hybrid scheduling of messages

In [10] and [11] we have introduced and evaluated a hybrid mechanism to schedule hard, soft, and non real-time messages in a CAN bus, under anticipated fault and load conditions. This method can be applied for message scheduling on every network with a priority-based basic medium access control. In following, the hybrid scheduling mechanism is described.

4.1. Scheduling of hard real-time messages

For hard real-time messages, the deadline is guaranteed by reserving the bus in a time slot (Figure 3). The reserved time slots are entered into a calendar. The message scheduling calendar is contained in all nodes of the system. This provides global knowledge on the bus schedule, which is important when the system is faced with the "babbling idiot" problem [7]. Task scheduling calendars are maintained locally on each node.

In order to merge static and dynamic scheduling of hard and soft real-time messages on the bus, we apply a tri-stage priority scheme, which is similar to the dual-priority approach [2]. Due to this scheme, a hard real-time message gains the highest possible priority at the beginning of its reserved time slot. Within its reserved time-slot, a hard real-time message is sent according to CSMA without waiting[1]. This results in a collision-free transmission, because all other messages have an additional waiting time. The minimum time-slot length de-

[1] Of course there is a minimum inter-frame spacing between two consecutive messages. This bus idle period must be additionally taken into account for all kinds of messages.

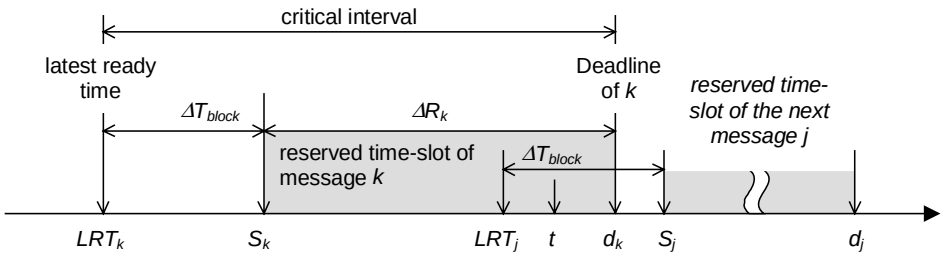

Figure 3. The reserved time-slot and critical interval of hard real-time messages

pends on the message length, bus speed, and the number of consecutive transmission failures to be tolerated. If, however, the laxity of a hard real-time message becomes negative, then the message is dropped, and exception is raised at the sender site.

Due to the non-preemptive nature of message transmission on CSMA busses, every message may be delayed by one message, which is started before its ready time. Thus, for the bus scheduling, we define ΔT_{block} as the longest possible message transmission time. A hard real-time message k (with a reserved time slot beginning at S_k) must be ready before S_k - ΔT_{block} (Figure 3). In order to prevent being blocked within its reserved time-slot by the transmission of any other message, a hard real-time message k must win the bus after S_k - ΔT_{block}. For this reason, between S_k - ΔT_{block} and S_k, k is transmitted with a non-zero waiting time ΔC_{min}, which is the minimum time required for the carrier signal on the bus to be propagated and detected by all nodes.

Consider the situation illustrated in Figure 3. If at time t the bus becomes idle, and hard real-time messages k and j and a soft real-time message l compete for the bus, then k is transmitted with no waiting, and finishes before d_k, because its laxity is non-negative at its transmission start time. If k has already been transmitted, then j is transmitted with the waiting time ΔC_{min}, and l is transmitted with a waiting time $\Delta C_l \in [2{*}\Delta C_{\text{min}} \ldots M{*}\Delta C_{\text{min}}]$, where M > 2. Thus, j is transmitted before l, and no collision occurs.

The proposed scheduling approach for hard real-time communication requires access to a global time reference with bounded inaccuracy. To guarantee that messages k and j do not collide, their senders must agree that t is inside of the reserved time-slot of k, and outside of the reserved time-slot of j. This agreement is guaranteed by leaving a gap between different reserved time-slots. The lower the clock accuracy, the larger the minimum gap between two subsequent time slots in the global bus schedule must be (Figure 4). Thus the required gap between reserved time-slots of different nodes can be bounded by synchronizing the local clocks of all non-faulty nodes [5]. In contrast to the bus scheduling, in case of local tasks the gap between reserved time-slots is independent from the clock accuracy, and only depends on task switching overhead.

Based on the previous discussion, we show that following requirements are sufficient for the correct scheduling of hard real-time messages on the bus:

(R1) for each possible occurrence of a hard real-time message k, an exclusive time-slot $[S_k \ldots d_k]$ is reserved, with d_k being its transmission deadline,

(R2) the length ($\Delta R_k = d_k - S_k$) of the reserved time-slot of a hard real-time message k is greater or equal to the worst-case transmission time of the message k, including all overheads for necessary retransmissions under anticipated fault conditions, and $\Delta R_k \geq \Delta T_{\text{block}}$.

(R3) the gap between any two different reserved time-slots for hard real-time messages is greater than the maximum clock-difference between any two correct nodes in the system,

(R4) every hard real-time message k is ready for transmission at its latest ready time $LRT_k = S_k - \Delta T_{\text{block}}$. This means that the laxity of every hard real-time message at its ready-time allows the longest possible message of the system to be transmitted first,

(R5) between its latest ready time and the beginning of its reserved time-slot, the waiting period of a hard real-time message is ΔC_{min}, and within the reserved time-slot the message is sent without waiting.

(R6) Soft real-time messages are always transmitted with a waiting period which is in the range $[2{*}\Delta C_{\text{min}} \ldots M{*}\Delta C_{\text{min}}]$, with M > 2 (cf. section 4.2),

(R7) before its latest ready time, the waiting period of a hard real-time message is a random period between $(M+1){*}\Delta C_{\text{min}}$ and $\Delta C_{\text{max}} \geq (M+2){*}\Delta C_{\text{min}}$ (to assure that it has a lower priority than soft real-time messages),

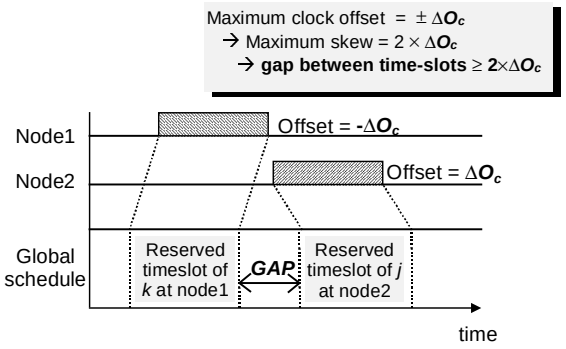

Figure 4. The gap between reserved time-slots due to clock inaccuracy

(R8) if the laxity of a hard real-time message is negative, then the message is dropped by its sender,

(R9) non real-time messages have waiting periods ≥ $\Delta C_{max} + \Delta C_{min}$ (cf. section 4.3).

Claim 1. If the requirements R1 through R9 are fulfilled in a CSMA bus, then every hard real-time message will be transmitted timely under anticipated fault conditions.

Proof. Assume (Figure 3) that a hard real-time message j exists with the deadline d_j. Due to R1 and R2 a time-slot $[S_j = d_j - \Delta R_j \ldots d_j]$ is reserved for j. Due to R4, j is ready for transmission at $LRT_j = d_j - \Delta R_j - \Delta T_{block}$. If j is not successfully transmitted until LRT_j, then it will compete for the bus at $t \in [LRT_j \ldots S_j)$. (let's label this situation as C1-A)

Due to R6, j wins the bus against every soft and non real-time message. Due to R1, R2, and R3, if another sender is trying to send a hard real-time message k, then it agrees that t is not within the interval $[LRT_k \ldots S_k)$, thus the waiting period of k is not ΔC_{min}, and k and j do not collide. If $t < LRT_k$, then k is transmitted with a waiting period $\geq (M+1)*\Delta C_{min}$, and j wins the bus at t. If $t \geq S_k$, then $t \in [S_k \ldots d_k]$, and k is transmitted without waiting, hence winning the bus. However, in this case k will be finished until d_k, because hard real-time messages with a negative laxity are dropped by the sender due to R8. Thus j will compete for the bus at t', where $t' \leq d_k$ as seen by the sender of k, and due to R3, $t' < S_j$ as seen by the sender of j. Again, j is in the situation C1-A, just substituting t by t'. By iteration, j will win the bus and start transmission before S_j, and it will win every bus arbitration during $[S_j \ldots d_j]$. since j gains the bus access for a period $> \Delta R_j$, it will be transmitted successfully until d_j under anticipated fault conditions. □

4.2. Scheduling soft real-time messages

For the soft real-time communication, the presented mechanism does not guarantee a deadline. However, this approach applies VTCSMA-D on the bus, to achieve EDF scheduling of soft real-time communication. For this reason, the waiting period of a soft real-time message l at the time t is determined as:

$$\Delta C_l(t) = \min\left\{M * \Delta C_{min}, \max\left\{2 * \Delta C_{min}, \frac{d_l(t) - t}{\eta}\right\}\right\}$$

The deadline of a message x at the time t – denoted as $d_x(t)$ – is determined based on the maximum transmission time of x and the estimation of the current laxity $L_A(t)$ of the associated real-time transaction A, i.e. $d_x(t) = t + MT_x + L_A(t)$. After a task or message x is completed at t', the laxity of the real-time transaction is recalculated as $L_A(t') = d_x(t) - t'$. If a task receives several messages, then it locally calculates the laxity of the real-time transaction as the minimum laxity inherited from any of those messages. At the ready time t_o of the beginning task of a real-time transaction A, the laxity $L_A(t_o)$ is estimated as $d_A - t_o - MET_A$, where d_A is the deadline of A, and MET_A is the maximum execution time of A. MET_A can be calculated at the design time, by applying the method described in [14] at a higher level, based on the maximum execution times of tasks and maximum transmission times of messages (including the anticipated time redundancy), and the graph representation of A (cf. Figure 1).

According to this scheme, soft real-time messages are scheduled by EDF, which is known to be optimal [9].

4.3. Scheduling non real-time messages

Non real-time messages are assigned fixed waiting periods, because the importance of a non real-time message does not change by the passage of time. In order to fulfil the requirement that non real-time messages only use the bus in absence of real-time messages, the waiting period of non real-time messages is randomly chosen from a range $[\Delta C_{max} + \Delta C_{min} \ldots \Delta C_{NRT}]$, where $\Delta C_{NRT} \geq \Delta C_{max} + 2*\Delta C_{min}$.

4.4. Dealing with faults

In order to guarantee timely hard real-time message transfer in the presence of faults, redundant communication resources must be provided. Space redundancy would require a second bus, which is too expensive for many application areas. When a fault model with only *crash* and *omission failures* of the communication system is assumed, time redundancy can be applied instead of

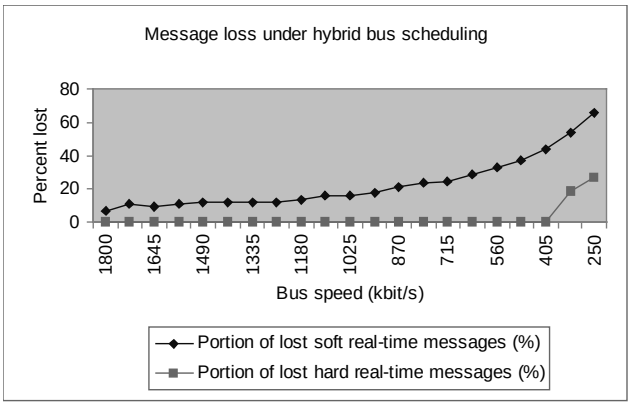

Figure 5. Simulation of the hybrid bus scheduling algorithm

space redundancy. This means that several subsequent transmissions have to be scheduled. This application of time redundancy is similar to the strategy used in early versions of the Time-Triggered Protocol [4].

5. Simulation results

We simulated a set of 42 periodic and sporadic hard and soft real-time messages, which are sent by 7 computing nodes. A list of these messages is contained in appendix 1. The rate of the virtual clock of the VTCSMA algorithm was set to η=8. The total data load caused by the messages, including headers, CRC, inter-frame spacing, etc. is about 0.466 Mbit/s. We simulated the scenario at different bus loads by varying the bus speed at constant data load. This is equivalent to varying the data load (e.g. for redundant transmissions) at constant bus speed. In our simulation, late soft real-time messages are dropped, because otherwise, at an average bus load over 100% the percentage of late soft real-time messages would depend on the length of the simulated period, and grow to 100% at large simulation periods. Note that Zhao and Ramamritham also dropped late soft real-time messages in their simulations [16].

In order to simulate the worst case load for the scheduling, the message arrival times have been adjusted such that deadlines of soft real-time messages are equal, hence resulting in the maximum number of collisions, and temporary overload conditions even in low average bus loads. Although very pessimistic, this case must be taken into account to show whether message loss is possible or not.

We simulated the bus scheduling for a period of 10^8 bit-times (corresponding to 55 to 400 seconds, depending on the bus speed). Appendix 2 contains the simulation results as numbers.

Our simulation (cf. Figure 5) showed that the hybrid scheduling mechanism is superior to the pure EDF scheduling based on VTCSMA-D, because in the worst case of burst equal deadlines, VTCSMA-D results in timing failure due to temporary overload conditions even under an average bus load of about 25%. In contrast, the calendar-based scheduling could guarantee timely message delivery even under average loads above 100%. At the bus speed of 0.405 Mbit/s (about 115% average bus load), only 0.07% of hard real-time messages were lost. At lower bus speeds, more hard real-time messages were lost. The loss of hard real-time messages at low bus speeds was because some hard real-time messages could not reserve any time-slot, thus they could not be scheduled correctly. If the overload condition is temporarily produced by a burst of soft real-time messages, then it does not lead to the timing failure of the hard real-time communication.

Compared to the TDMA technique, our scheme achieves a higher bus utilization by transmitting soft real-time and non real-time messages within reserved time-slots, whenever hard real-time messages do not need transmission.

6. Conclusion

In order to guarantee timely completion of hard real-time messages in a CSMA network, we have introduced a calendar-based scheduling mechanism, which coexists with the EDF scheduling used for soft real-time communication. In contrast to pure EDF scheduling, where resource conflicts may occur due to equal deadlines, the presented approach regulates the deadlines of hard real-time messages by reserving transmission times in a global calendar. Thus, the timeliness of hard real-time communication is achieved despite overload failures by guaranteeing exclusive access right to the bus during the reserved time-slots.

Although resources are exclusively reserved for periodic and sporadic hard real-time messages, whenever a sporadic hard real-time message is not transmitted with its highest anticipated frequency, its associated communication resources are utilized by soft real-time and non real-time messages.

Our simulations have shown, that in the worst case where all deadlines are equal, the pure EDF scheduling is not capable of scheduling messages under high bus loads, and performs poorly because of a high number of collisions. In contrast, our approach schedules hard real-

time communication correctly, even in overload situations.

7. References

[1] A. Damm, J. Reisinger, W. Schwabl, and H. Kopetz, "The Real-Time Operating System of MARS", *ACM Operating Systems Review 23(3):141-157, July 1989*.

[2] R. Davis, "Dual Priority Scheduling: A Means of Providing Flexibility in Hard Real-time Systems", *Report No. YCS230, University of York, UK, May 1994*.

[3] G. Fohler and C. Koza, "Heuristic Scheduling for Distributed Real-Time Systems", *Research Report No. 12/90, Inst. für Techn. Informatik, Techn. University of Vienna, 1990*.

[4] H. Kopetz and W. Merker, "The Architecture of MARS", *Proc. of 15th Fault Tolerant Computing Symposium, pp. 274-279, Ann Arbor, Michigan, 1985*.

[5] H. Kopetz and W. Ochsenreiter, "Clock Synchronization in Distributed Real-Time Systrems", *IEEE Trans. on Computers 36(8):933-940, Aug. 1987*.

[6] H. Kopetz and G. Grünsteidl, "TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems", *Research Report No. 12/92, Inst. für Techn. Informatik, Techn. University of Vienna, 1992*.

[7] A. Krüger and H. Kopetz, "A Network Controller Interface for a Time-Triggered Protocol", *1995 SAE Symposium on Future Transportation Electronics: Multiplexing and In-Vehicle Networking.*

[8] S.T. Levi, S.K. Tripathi, S.D. Carson, and A.K. Agrawala, "The MARUTI Hard Real-Time Operating System", *ACM Operating Systems Review 23(3):90-105, July 1989*.

[9] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment", *J. ACM 20(1):46-61, 1973*.

[10] M.A. Livani, J. Kaiser, and W. Jia, "Scheduling Hard and Soft Real-Time Communication in the Controller Area Network (CAN)", *23rd IFAC/IFIP Workshop on Real Time Programming (WRTP98), Shantou, China, 1998*.

[11] M.A. Livani and J. Kaiser, "Evaluation of a Hybrid Real-time Bus Scheduling Mechanism for CAN", *Lecture Notes in Computer Science 1586 (Jose Rolim et al. Eds.), pp. 425-429, Springer Verlag Berlin, 1999*.

[12] M.L. Molle and L. Kleinrock, "Virtual Time MSMA: Why two clocks are better than one", *IEEE Transactions on Communications 33(9), Sep. 1985*.

[13] P. Puschner and Ch. Koza, "Calculating the Maximum Execution Time of Real-Time Programs", *J. Real Time Systems 1(2):159-176, 1989*.

[14] P. Puschner and A. Schedl, "Computing Maximum Task Execution Times – A Graph-Based Approach", *J. Real Time Systems 13(1):67-91, 1997*.

[15] SAE, "Class C Application Requirement Considerations", *SAE Technical Report J2056/1, June 1993.*

[16] W. Zhao and K. Ramamritham, "Virtual Time CSMA Protocols for Hard Real-Time Communcations", *IEEE Tr. Software Eng. 13(8), 1987*.

Appendix 1. The message set used for simulations

Following set of messages was used for simulations. It was obtained by modifying a benchmark for safety critical vehicle control (Class C) applications [15].

Signal (message) description	*bits*	*jitter*	*period*	*Kind*	*deadline*	*sender*	*receiver*
Battery high rate	32	0.6	100	PS	100	Battery	V/C
Battery low rate	24	1.0	1000	PH	1000	Battery	V/C
Accelerator Position	8	0.1	5	PH	5	Driver	V/C
Brake high rate	16	0.1	5	PH	5	Brakes	V/C
Transaxle Lubrication Pressure	8	0.2	100	PS	100	Trans	V/C
Transaction Clutch Line Pressure	8	0.1	5	PS	5	Trans	V/C
Vehicle Speed	8	0.8	100	PH	100	Brakes	V/C
Hi/Lo Contactor Open Close	4	0.1	50	H	5	Battery	V/C
Key Switch Run	1	0.2	50	S	20	Driver	V/C
Key Switch Start	1	0.3	50	S	20	Driver	V/C
Accelerator Switch	2	0.4	50	S	20	Driver	V/C
Brake Switch	1	0.3	20	S	20	Brakes	V/C
Emergency Brake	1	0.5	50	H	20	Driver	V/C
Shift Lever (PRNDL)	3	0.6	50	S	20	Driver	V/C
Motor/Trans Over Temperature	2	0.3	1000	H	1000	Trans	V/C
Speed Control	3	0.7	50	S	20	Driver	V/C
12V Power Ack Vehicle Control	1	0.2	50	S	20	Battery	V/C
12V Power Ack Inverter	1	0.3	50	S	20	Battery	V/C
12V Power Ack I/M Control	1	0.4	50	S	20	Battery	V/C
Brake Mode (Parallel/Split)	1	0.8	50	S	20	Driver	V/C
SOC Reset	1	0.9	50	S	20	Driver	V/C
Interlock	1	0.5	50	S	20	Battery	V/C
Reverse and 2nd Gear Clutches	2	0.5	50	S	20	V/C	Trans
V/C high rate	32	0.1	5	PH	5	V/C	Battery
V/C low rate	16	1.6	1000	PH	1000	V/C	Battery
DC/DC Converter Current Control	8	0.6	50	S	20	V/C	Battery
12V Power Relay	1	0.7	50	S	20	V/C	Battery
Brake Solenoid	1	0.8	50	H	20	V/C	Brakes
Backup Alarm	1	0.9	50	H	20	V/C	Battery
Warning Lights	7	1.0	50	H	20	V/C	Ins.
Key Switch	1	1.1	50	S	20	V/C	I/MC
Main Contactor Close	1	0.3	50	S	20	I/MC	V/C
I/M C high rate	16	0.1	5	PH	5	I/MC	V/C
FWD/REV	1	1.2	50	S	20	V/C	I/MC
FWD/REV Ack.	1	0.4	50	S	20	I/MC	V/C
Idle	1	1.3	50	S	20	V/C	I/MC
Inhibit	1	0.5	50	S	20	I/MC	V/C
Shift in Progress	1	1.4	50	S	20	V/C	I/MC
Inverter Temperature Status	2	0.6	50	S	20	I/MC	V/C
Shutdown	1	0.7	50	H	20	I/MC	V/C
Status/Malfunction (TBD)	8	0.8	50	H	20	I/MC	V/C
Main Contactor Acknowledge	1	1.5	50	S	20	V/C	I/MC

1) Kind: PH = periodic hard RT; H = sporadic hard RT; PS = periodic soft RT; S = sporadic soft RT.
2) Period, deadline, and jitter are given in milliseconds.

Appendix 2. Simulation results in form of numbers

Bus speed (kilo bit/s)	Portion of lost soft real-time messages (%)	Portion of lost hard real-time messages (%)
1800	6,9853	0
1722,5	10,9244	0
1645	9,1582	0
1567,5	11,104	0
1490	11,8501	0
1412,5	11,8353	0
1335	11,9387	0
1257,5	12,1989	0
1180	13,507	0
1102,5	15,7562	0
1025	16,0034	0
947,5	17,7245	0
870	20,9768	0
792,5	23,7382	0
715	24,4598	0
637,5	28,9398	0
560	32,9869	0
482,5	37,0487	0
405	43,4115	0,0715
327,5	54,1286	18,869
250	66,0189	26,8459