Hierarchical Scheduling of CAN using Server-Based Techniques

Thomas Nolte, Mikael Nolin, and Hans Hansson
Mälardalen Real-Time Research Centre
Deptartment of Computer Science and Engineering
Mälardalen University, Västerås, SWEDEN
Email: {thomas.nolte, mikael.nolin, hans.hansson}@mdh.se

Abstract

Server-based scheduling of CAN has been proposed as a way of fair scheduling of the Controller Area Network (CAN). By separating streams of messages by using network access servers (N-Servers), scheduling is performed at three layers where native CAN message arbitration is the scheduling at the lowest level. On top of this is the server-based scheduling to separate different streams of messages within the system. Looking at a single N-Server, several streams might share one server. Hence, scheduling is performed at the third level every time the N-Server is being scheduled for message transmission. Here different queuing policies play a role in the scheduling performed.

This paper discusses the hierarchical scheduling of CAN, as a way of fair separation of message streams while providing a flexible core mechanism, the server-based scheduling of CAN.

1 Introduction

In optimising the design of a real-time communications system it is important both to guarantee the timeliness of periodic messages and to minimize the interference from periodic traffic on the transmission of aperiodic messages. Therefore, we propose the usage of *server-based scheduling techniques*, providing bandwidth isolation and efficient handling of streams of periodic and aperiodic traffic. The need for this kind of scheduled real-time communication network is high in applications that have requirements on flexibility, both during development for assigning communication bandwidth to different applications, and during run-time to facilitate dynamic addition and removal of system components, e.g., open systems. Examples of server-based scheduling techniques for CPU scheduling are the Total Bandwidth Server (TBS) [15], and the Constant Bandwidth Server (CBS) [1]. In this paper we are working with a distributed CAN network, where a dynamic priority version of the Polling Server (PS) [7] is used.

The paper is organised as follows: In Section 2 the Controller Area Network (CAN) is presented together with different existing scheduling policies. In Section 3 Server-based scheduling of CAN is presented. In Section 4 hierarchical scheduling of CAN is presented, and finally Section 5 concludes the paper.

2 The Controller Area Network (CAN)

The Controller Area Network (CAN) [4] is a broadcast bus designed to operate at speeds of up to 1Mbps. CAN is extensively used in automotive systems, as well as in other applications. CAN is a collision-avoidance broadcast bus, which uses deterministic collision resolution to control access to the bus (so called CSMA/CA). CAN guarantees that the highest priority active frame will be transmitted. Hence, CAN behaves like a priority-based queue.

2.1 Scheduling of CAN

Looking at the real-time research community for different scheduling policies, three major types are found namely priority-driven (e.g., Fixed-Priority Scheduling (FPS) or Earliest Deadline First (EDF)) [8], time-driven (table-driven) [6, 5], and share-driven [13, 16].

For CAN, the most natural scheduling method is FPS since it is the policy used by the CAN protocol. Analysis, like the FPS response-time tests to determine the schedulability of CAN message frames, have been presented by Tindell *et al.* [17].

Several methods for dynamic-priority type of scheduling have been proposed for CAN. By manipulating the message identifier, and therefore changing the message priority dynamically, several approaches for EDF type of scheduling have been presented [9, 10, 19].

Table-driven, or time-driven, scheduling of CAN is provided by, e.g., TT-CAN [18] and FTT-CAN [3]. Flexible Time-Triggered CAN (FTT-CAN), presented by Almeida *et al.*, supports priority-driven scheduling in combination with time-driven scheduling. It has also been shown how to schedule periodic messages according to EDF using FTT-CAN [14].

Share-driven scheduling of CAN has been presented in our earlier work on server-based scheduling of CAN [11, 12]. By providing the option of share-driven scheduling of CAN, designers are given more freedom in designing a distributed real-time system where the nodes are interconnected with a CAN-bus.

3 Server-based Scheduling of CAN

In server-based scheduling of CAN, bandwidth is allocated to users of the network by the usage of servers called network access servers (*N-Servers*). Each node has one or more N-Servers allocated to it. The N-Servers have exclusive associated bandwidth in terms of capacity, C, and a period time, T. Moreover, all N-Servers have an associated deadline in order to be scheduled for message transmission. Time is divided into Elementary Cycles (ECs), similar to the FTT-CAN [3]. The length of an EC is denoted T_{EC}. Hence, the N-Server period, T, is an integer multiple of T_{EC}.

All N-Servers have a local message queue, in which all its user messages are stored. A *user* is a stream of messages, e.g., the sending of messages by an application, and can be of either periodic, sporadic, or aperiodic nature.

The scheduling is performed at a specialised master node, called the *M-Server*. The M-Server keeps all information regarding the N-Servers in the system. Hence the M-Server has all the N-Server's parameters. The M-Server is updating the N-Server deadlines according to the scheduling policy in use.

As soon as an N-Server is being scheduled for message transmission, the N-Server selects a message from its local message queue. Since each N-Server has exclusive right to a share of the total system bandwidth, all users sharing an N-Server will share this portion of bandwidth. Hence, depending on the type of queue used at the N-Server, e.g., FIFO or priority-based, different guarantees of timeliness can be offered. Suppose a priority-based queue is used, then the users will experience a service, in terms of timeliness, similar to the one of an exclusive network, essentially with the only difference in the lower bandwidth offered. Hence, the timely behaviour will be, compared to an exclusive CAN network, divided by C/T, i.e., the server's share. A variant of the response-time analysis for fixed priority systems could be used to calculate the timing properties. This is explained in detail in Section 4.

The M-Server and all N-Servers are sending their messages to their corresponding CAN controller, where all messages are scheduled according to the native CAN message arbitration mechanism.

3.1 Scheduling Mechanism

The server-scheduling mechanism is depicted in Figure 1.

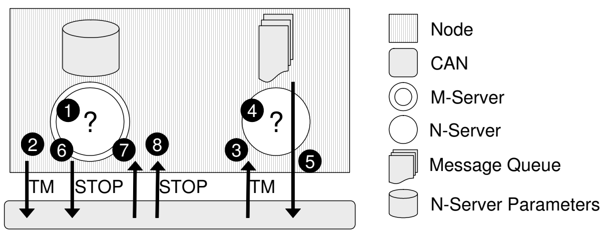

Figure 1. Server-scheduling mechanism for CAN.

The scheduling is performed at the M-Server according to the Earliest Deadline First (EDF) policy (Figure 1:1, i.e., (1) in Figure 1). A schedule is created containing the N-Servers with the earliest deadlines, filling up one EC. As this is done, the schedule is put into a trigger message (TM) (depicted in Figure 2) and multicasted to all the N-Servers in the system (Figure 1:2).

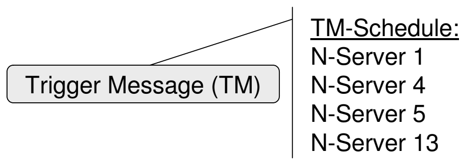

Figure 2. Example of a Trigger Message (TM) and its contents.

When the N-Servers receive the TM (Figure 1:3), they will read it to see whether or not they are allowed to send a message (Figure 1:4). If they are, their message is immediately queued for transmission (if existing) at the CAN controller (Figure 1:5). Otherwise, if not scheduled, the node has to wait for the next TM to see if it was scheduled that time.

In order to terminate the EC, the M-Server is also sending a STOP message to itself (Figure 1:6). This is done right after the TM is sent. The STOP message is of lowest priority possible, acting as an indicator for when all the nodes that were allowed to send a message within the active EC actually have sent their messages. If the M-Server receives the STOP message, all nodes that were allowed to send messages within the EC have already sent their messages. This since the STOP message, with its low priority, is the last message to be sent within the EC. An example EC is depicted in Figure 3.

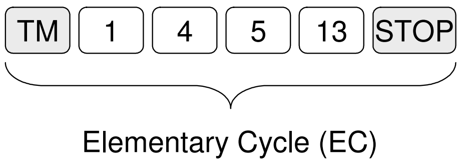

Figure 3. Elementary Cycle (EC) containing messages from 4 N-Servers.

The M-Server is always reading all the messages that are sent on the CAN bus (Figure 1:7), i.e., the M-Server is polling the bus. This in order to update its server variables based on the actual traffic sent on the bus. Since servers are scheduled for message transmission even though they might not always have any messages to send, this has to be taken care of by updating the server parameters accordingly. There are different ways of updating the server-parameters in the case when the server did not send a message. Depending on how the server-parameters are updated the server will have different real-time characteristics [11].

When the M-Server reads the STOP message (Figure 1:8), the EC is terminated, and the next EC is initiated based on the updated server-variables.

Using the server-based concept, servers and users can potentially join and leave the system arbitrary as long as the total utilisation, or bandwidth demand, in the system by all the servers is less or equal to the theoretical maximum given by (1) below. Note that the joining and leaving of users does not affect other users (in terms of bandwidth) than those sharing the same server, due to the bandwidth isolation between different N-Servers.

3.2 Bandwidth

Note that this scheduling mechanism limits the available bandwidth on the network. The theoretical maximum network utilisation is expressed by

$$\sum_{\forall s}\left(\frac{M}{S \times T_s}\right) \leq 1 - \left(\frac{TM + STOP + S \times T_{sched}}{S \times T_{EC}}\right) \tag{1}$$

where s is an N-Server in the system, M is the length of a message in bits (typically worst-case which is 135 bits), S is the network speed in bits/second, T_s is the period of the N-Server. TM and STOP are the sizes of the TM and STOP messages in bits, typically 135 and 55 bits, T_{sched} represents the computational overhead in time (seconds) of updating the N-Server deadlines and encoding the next TM after receiving the STOP message, and T_{EC} is the length of the EC in time (seconds).

4 Hierarchical Scheduling of CAN

At the N-Server, user messages are queued in a local message queue. This message queue can, for instance, be of either FIFO or priority based nature. In the case of a priority based message queue the behaviour of the network, from a user point of view, is as an exclusive CAN bus that has a bandwidth of essentially C/T of the whole bandwidth capacity. Hence, different scheduling policies are performed at different levels as depicted in Figure 4.

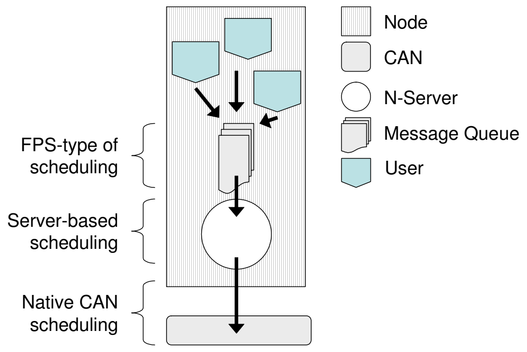

Figure 4. Hierarchical Scheduling of CAN.

4.1 Timely Behaviour

Depending on the server-parameter updating, different timing behaviour is achieved. Using the PS^2-CAN of [11] with $C = 1$ (i.e., the N-Server capacity is one message in each N-Server period T_s), for a given message stream i, the worst-case message response-time is as follows

$$R_i = q_i + T_{EC} \tag{2}$$

where T_{EC} is the length (in time) of an elementary cycle (to cover for the time granularity due to EC-scheduling), and q_i represents the effective queuing time given by

$$q_i^n = \left(2 + \sum_{j \in hp(i)} \left\lceil \frac{q_i^{n-1}}{T_j} \right\rceil \right) \times T_s \tag{3}$$

where "2" is the worst case response-time (in multiples of N-Server periods) for a single message using PS^2-CAN according to [11], and the summation represents interfering higher priority user message streams sharing the same N-Server, where $hp(i)$ is the set of these streams with priority higher than stream i, and T_j is their corresponding period-time. Finally, T_s is the period-time of the N-Server. Note that $T_j \geq T_s$. The first approximation is $q_i^0 = 0$. A solution is reached either when $q_i^{n+1} = q_i^n$, or when q_i^n exceeds its message deadline or period.

5 Summary

In this paper we have presented hierarchical scheduling of CAN using server-based techniques. We have presented the scheduling mechanisms performed at different levels, and we have shown that this hierarchical scheduling of CAN provides several advantages, e.g., bandwidth isolation and flexibility. Moreover, we have presented a timing analysis from a user point of view.

The presented mechanism is suitable for open systems, where users can join and leave the system at any time. However, an admission control needs to be implemented. This can easily be done by implementing the admission control in the M-Server together with the usage of dedicated N-Servers with a small amount of bandwidth on each node for the admission control mechanism. This would be an interesting future work.

Another interesting future work would be to see how the work presented in [2] relates to the work in this paper, and the possibility of a joint effort in this topic.

Acknowledgements

The work presented in this paper was supported by the Swedish Foundation for Strategic Research (SSF) via the research programmes ARTES and SAVE, the Swedish Foundation for Knowledge and Competence Development (KK-stiftelsen), and Mälardalen University.

References

[1] L. Abeni. Server Mechanisms for Multimedia Applications. Technical Report RETIS TR98-01, Scuola Superiore S. Anna, Pisa, Italy, 1998.

[2] L. Almeida. Response-time Analysis and Server Design for Hierarchical Scheduling. In *WIP Proceedings of* 24th *IEEE Real-Time Systems Symposium (RTSS'03)*, pages 121–124, Cancun, Mexico, December 2003.

[3] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN Protocol: Why and How. *IEEE Transaction on Industrial Electronics*, 49(6), December 2002.

[4] CAN. Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed Communication. *International Standards Organisation (ISO)*, ISO Standard-11898, Nov 1993.

[5] C. W. Hsueh and K. J. Lin. An Optimal Pinwheel Scheduler Using the Single-Number Reduction Technique. In *Proceedings of the* 17th *IEEE Real-Time Systems Symposium (RTSS'96)*, pages 196–205, Los Alamitos, CA, USA, December 1996. IEEE Computer Society.

[6] H. Kopetz. The Time-Triggered Model of Computation. In *Proceedings of the* 19th *IEEE Real-Time Systems Symposium (RTSS'98)*, pages 168–177, Madrid, Spain, December 1998. IEEE Computer Society.

[7] J. Lehoczky, L. Sha, and J. Strosnider. Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. In *Proceedings of* 8th *IEEE Real-Time Systems Symposium (RTSS'87)*, pages 261–270, San Jose, California, USA, December 1997. IEEE Computer Society.

[8] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):40–61, 1973.

[9] M. Livani and J. Kaiser. EDF Consensus on CAN Bus Access for Dynamic Real-Time Applications. In *Proceedings of the* 6th *International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'98)*, Orlando, Florida, USA, March 1998.

[10] M. D. Natale. Scheduling the CAN Bus with Earliest Deadline Techniques. In *Proceedings of the* 21st *IEEE Real-Time Systems Symposium (RTSS'00)*, pages 259–268, Orlando, Florida, USA, November 2000. IEEE Computer Society.

[11] T. Nolte, M. Nolin, and H. Hansson. Server-Based Real-Time Scheduling of the CAN Bus. In *Proceedings of* 11th *IFAC Symposium on Information Control Problems in Manufacturing (INCOM'04)*, Salvador, Brasil, April 2004.

[12] T. Nolte, M. Sjödin, and H. Hansson. Server-Based Scheduling of the CAN Bus. In *Proceedings of the* 9th *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03)*, pages 169–176, Calouste Gulbenkian Foundation, Lisbon, Portugal, September 2003.

[13] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[14] P. Pedreiras and L. Almeida. A Practical Approach to EDF Scheduling on Controller Area Network. In *Proceedings of the IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01) at the* 22nd *IEEE Real-Time Systems Symposium (RTSS'01)*, London, England, December 2001.

[15] M. Spuri, G. C. Buttazzo, and F. Sensini. Robust Aperiodic Scheduling under Dynamic Priority Systems. In *Proceedings of the* 16th *IEEE Real-Time Systems Symposium (RTSS'95)*, pages 210–219, Pisa, Italy, December 1995. IEEE Computer Society.

[16] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and G. Plaxton. A Proportional Share Resource Allocation Algoritm for Real-Time, Time-Shared Systems. In *Proceedings of* 17th *IEEE Real-Time Systems Symposium (RTSS'96)*, pages 288–299, Los Alamitos, CA, USA, December 1996. IEEE Computer Society.

[17] K. W. Tindell, A. Burns, and A. J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[18] TT-CAN. Road Vehicles - Controller Area Network (CAN) - Part 4: Time-Triggered Communication. *International Standards Organisation (ISO)*, ISO Standard-11898-4, December 2000.

[19] K. M. Zuberi and K. G. Shin. Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications. In *Proceedings of the* 1st *IEEE Real-Time Technology and Applications Symposium (RTAS'95)*, pages 240–249, Chicago, IL, USA, May 1995. IEEE Computer Society.