Providing Bandwidth Isolation on the Controller Area Network by Using Servers

Thomas Nolte, Mikael Sjödin, and Hans Hansson
Mälardalen Real-Time Research Centre
Department of Computer Science and Engineering
Mälardalen University, Västerås, SWEDEN
http://www.mrtc.mdh.se

Abstract

In this paper we present a new share-driven server-based method for scheduling messages sent over the Controller Area Network (CAN) [14, 4]. Share-driven methods are useful in many applications, since they provide both fairness and bandwidth isolation among the users of the resource. Our method is the first share-driven scheduling method proposed for CAN, and it is based on Earliest Deadline First (EDF), which allows higher utilization of the network than does CAN's native fixed-priority scheduling approach. We use simulation to show the performance and properties of server-based scheduling for CAN. The simulation results show that the bandwidth isolation property is kept, and they show that our method provides a Quality-of-Service (QoS), where virtually all messages are delivered within their deadline.

Today, distributed real-time systems become more and more complex and the number of micro controllers attached to CAN buses continue to grow. CAN's maximum speed of 1 Mbps remains, however, fixed; leading to performance bottlenecks. This bottleneck is further accentuated by the steadily growing computing power of CPUs. Studies have shown that CAN's priority mechanism allows for lower network utilization than the Earliest Deadline First (EDF) mechanism [8, 12]. Hence, in order to reclaim some of the scarce bandwidth forfeited by CAN's native scheduling mechanism, novel approaches to scheduling CAN are needed.

In optimising the design of a CAN-based communication system (and essentially any other real-time communication system) it is important to both guarantee the timeliness of periodic messages and to minimize the interference from this periodic traffic on the transmission of aperiodic messages. Therefore, in this paper we propose the usage of an EDF server based scheduling technique, which improves existing techniques since: (1) Fairness among the messages is guaranteed (i.e., "misbehaving" aperiodic processes cannot starve well-behaved processes), and (2) in contrast with other proposals, aperiodic messages are not sent "in the background" of periodic messages or in separate time-slots [11]. Instead, aperiodic and periodic messages are jointly scheduled using servers. This substantially facilitates meeting response-time requirements for both aperiodic and periodic messages.

In the real-time scheduling research community there exist several different types of scheduling. We can divide the classical scheduling paradigms into the following three groups:

1. *Priority-driven (e.g., FPS or EDF) [7].*
2. *Time-driven (table-driven) [6, 5].*
3. *Share-driven [10, 15].*

For CAN, priority-driven scheduling is the most natural scheduling method since it is supported by the CAN protocol, and FPS response-time tests for determining the schedulability of CAN message frames have been presented by Tindell et al. [16, 17, 18]. This analysis is based on the standard fixed-priority response-time analysis for CPU scheduling presented by Audsley et al. [3]. TT-CAN [13] provides time-driven scheduling for CAN, and Almeida et al. present Flexible Time-Triggered CAN (FTT-CAN) [1, 2], which supports priority-driven scheduling in combination with time driven-scheduling. FTT-CAN is presented in more detail below. The server-based scheduling presented in this paper provides the first share-driven scheduling approach for CAN. By providing the option of share-driven scheduling of CAN, designers are given more freedom in designing an application.

As a side effect, by using servers, the CAN identifiers assigned to messages will not play a significant role in the message response-time. This greatly simplifies the process of assigning message identifiers (which is often done in an ad-hoc fashion at an early stage in a project). This also allows migration of legacy systems (where identifiers cannot easily be changed) into our new framework.

The difference between our approach and existing methods is that we make use of server-based scheduling (based on EDF). Our approach allows us to utilize the CAN bus in a more flexible way compared to other scheduling approaches such as native CAN, and Flexible Time-Triggered communication on CAN (FTT-CAN). Servers provide fairness among the streams of messages as well as timely message delivery.

The strength of server-based scheduling for CAN, compared to other scheduling approaches, is that we can cope with streams of aperiodic messages. Aperiodic messages on native CAN would make it (in the general case) impossible to give any real-time guarantees for the periodic messages sharing the bus. In FTT-CAN the situation is better, since periodic messages can be scheduled according to EDF using the synchronous window of FTT-CAN, thus guaranteeing real-time demands. However, no fairness can be guaranteed among the streams of aperiodic messages sharing the asynchronous window of FTT-CAN. We have in [9] performed simulations where the bandwidth isolation property is verified.

One penalty for using the server method is an increase of CPU load in the master node, since it needs to perform the extra work for scheduling. Also, compared with FTT-CAN, we are sending one more message, the STOP message, which is reducing the available bandwidth for the system under heavy aperiodic load. However, the STOP message is of the smallest size possible and therefore it should have minimal impact on the system. However, if the CAN controller is able to detect when the bus is idle (and pass this information to the master node), we could skip the STOP message, and the overhead caused by our protocol would decrease (since this would make it possible to use our server-based scheduling without STOP-messages).

As we see it, each scheduling policy has both good and bad properties. To give the fastest response-times, native CAN is the best choice. To cope with fairness and bandwidth isolation among aperiodic message streams, the server-based approach is the best choice, and, to have support for both periodic and aperiodic messages (although no fairness among aperiodic messages) and hard real-time, FTT-CAN is the choice. Using server-based scheduling, we can schedule for unknown aperiodic or sporadic messages by guessing that they are arriving, and if we make an erroneous guess, we are not wasting much bandwidth. This since the STOP message, together with the arbitration mechanism of CAN, allow us to detect when no more messages are pending so that we can reclaim potential slack in the system and start scheduling new messages without wasting bandwidth.

References

[1] L. Almeida, J. Fonseca, and P. Fonseca. Flexible Time-Triggered Communication on a Controller Area Network. In *Proceedings of the Work-In-Progress Session of the* 19^{th} *IEEE Real-Time Systems Symposium (RTSS'98)*, Madrid, Spain, December 1998. IEEE Computer Society.

[2] L. Almeida, J. Fonseca, and P. Fonseca. A Flexible Time-Triggered Communication System Based on the Controller Area Network: Experimental Results. In *Proceedings of the International Conference on Fieldbus Technology (FeT'99)*, Magdeburg, Germany, September 1999.

[3] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[4] CAN Specification 2.0, Part-A and Part-B. CAN in Automation (CiA), Am Weichselgarten 26, D-91058 Erlangen. http://www.can-cia.de/, 2002.

[5] C.-W. Hsueh and K.-J. Lin. An Optimal Pinwheel Scheduler Using the Single-Number Reduction Technique. In *Proceedings of the* 17^{th} *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* 19^{th} *IEEE Real-Time Systems Symposium (RTSS'98)*, pages 168–177, Madrid, Spain, December 1998. IEEE Computer Society.

[7] 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.

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

[9] T. Nolte, M. Sjödin, and H. Hansson. Server-Based Scheduling of the CAN Bus. Technical report, ISSN 1404-3041 ISRN MDH-MRTC-99/2003-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, Sweden, April 2003.

[10] A. Parekh and R. 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.

[11] P. Pedreiras and L. Almeida. Combining Event-triggered and Time-triggered Traffic in FTT-CAN: Analysis of the Asynchronous Messaging System. In *Proceedings of the* 3^{rd} *IEEE International Workshop on Factory Communication Systems (WFCS'00)*, pages 67–75, Porto, Portugal, September 2000. IEEE Industrial Electronics Society.

[12] 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* 22^{nd} *IEEE Real-Time Systems Symposium (RTSS'01)*, London, England, December 2001.

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

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

[15] 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* 17^{th} *IEEE Real-Time Systems Symposium (RTSS'96)*, pages 288–299, Los Alamitos, CA, USA, December 1996. IEEE Computer Society.

[16] K. W. Tindell and A. Burns. Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks. Technical Report YCS 229, Dept. of Computer Science, University of York, York, England, June 1994.

[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] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). In *Proceedings of* 15^{th} *IEEE Real-Time Systems Symposium (RTSS'94)*, pages 259–263, San Juan, Puerto Rico, December 1994. IEEE Computer Society.