Effects of varying phasings of message queuings in CAN based systems

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

Abstract

This article presents and illustrates the effects on message response times by considering variations in phasings of message queuings in distributed system using the CAN bus. Traditional worst-case analysis is based on very pessimistic assumptions. This may be correct from a hard real-time perspective, but from a system perspective where reliability is also an issue, this may lead to an unnecessary costly and over-designed system. The worst-case analysis assumes worst-case phasings of message queuings, and that messages are constantly queued with highest possible frequency. In this paper we will investigate the level of introduced pessimism by looking into the effects of relaxing these assumptions. We will investigate the pessimism by simulating these systems, both without taking into account the effects of phasing, and with. We show the actual effect with a simple case-study. The motivation for this paper is to underline the pessimism introduced by not considering phasings. Simulation results show the pessimism by not considering phasings.

1. Introduction

In the past 20 years or so, real-time researchers have extended schedulability analysis to a mature technique which for non-trivial systems can be used to determine whether a set of tasks executing on a single CPU or in a distributed system will meet their deadlines or not [1][2][9] [13]. The core of this analysis is to investigate whether deadlines are met in a worst case scenario or not. If this worst case actually will occur during execution, or if it is likely to occur, is not normally considered.

In contrast with schedulability analysis, reliability modelling involves study of fault models, characterisation of distribution functions of faults and development of methods and tools for composing these distributions and models in estimating an overall reliability figure for the system.

The separation of deterministic (0/1) schedulability analysis and stochastic reliability analysis is a natural simplification of the total analysis. This because the deterministic schedulability analysis unfortunately is quite pessimistic since it assumes that a missed deadline in the worst case is equivalent to always missing the deadline whereas the stochastic analysis extend the knowledge of the system by telling how often a deadline is violated. Furthermore the failure semantics could be extended allowing the system to miss some deadlines and still not classify it as a failure. There are many other sources of pessimism in the analysis, including considering worst-case execution times, the usage of pessimistic fault models, and, what we will investigate in this paper, worst-case phasings of executions.

In our previous work [8], we have proposed a model for calculating worst-case latencies of Controller Area Network (CAN) [6] frames (messages) under error assumptions. This model is pessimistic, in the sense that there are systems that the analysis determine unschedulable, even though deadlines will only be missed in extremely rare situations with pathological combinations of errors. In [4][5] we have reduced the level of pessimism by introducing a better fault model, and in [3] we also consider variable phasings between message queuings, in order to make the model more realistic. In [7] we reduced the pessimism introduced by the worst-case analysis of CAN message response-times, by using bit-stuffing distributions instead of the traditional worst-case frame sizes.

In this paper we will focus on investigating the actual effects caused by phasings. The motivation for this is that we believe phasings is a source of pessimism not normally catered for. We will investigate the source of phasings, and then, investigate how different parameters related to phasings effect the system. This paper will mainly consider the phasings caused by varying the response times of the tasks queuing frames.

The outline of the article is as follows. Section 2 presents the Controller Area Network, and Section 3 describes the classical response time analysis for CAN. Section 4 talks about the effects of phasings and in Section 5 we show these in case-study. Finally Section 6 concludes the paper and discusses future work.

2. The Controller Area Network

The Controller Area Network (CAN) [6] is a broadcast bus designed to operate at speeds of up to 1 Mbps. Data is transmitted in frames containing between 0 and 8 bytes of data and 47 control bits. Among those control bits there is an 11-bit identifier associated with each frame. The identifier is required to be unique, in the sense that two simultaneously active frames originating from different sources must have distinct identifiers. The identifier serves two purposes: (1) assigning a priority to the frame, and (2) enabling receivers to filter frames. CAN is a collision-detect broadcast bus, which uses deterministic collision resolution to control access to the bus. The basis for the access mechanism is the electrical characteristics of a CAN bus: if multiple stations are transmitting concurrently and one station transmits a ’0’ then all stations monitoring the bus will see a ’0’. Conversely, only if all stations transmit a ’1’ will all processors monitoring the bus see a ’1’. During arbitration, competing stations are simultaneously putting their identifiers, one bit at the time, on the bus. By monitoring the resulting bus value, a station detects if there is a competing higher priority frame and stops transmission if this is the case. Because identifiers are unique within the system, a station transmitting the last bit of the identifier without detecting a higher priority frame must be transmitting the highest priority queued frame, and hence can start transmitting the body of the message.

3. Response Time Analysis of CAN

Tindell et al. [10] [11] [12] present analysis to calculate the worst-case latencies of CAN frames. This analysis is based on the standard fixed priority response time analysis for CPU scheduling [1].

Calculating the response times requires a bounded worst case queuing pattern of frames. The standard way of expressing this is to assume a set of traffic streams, each generating frames with a fixed priority. The worst-case behaviour of each stream, in the sense of network load, is to assume that each frame is periodically queued. In analogue with CPU scheduling, we obtain a model with a set $\mathcal{S}$ of streams (corresponding to CPU tasks). Each $S_i \in \mathcal{S}$ is a triple $< P_i, T_i, C_i >$, where P_i is the priority (defined by the frame identifier), T_i is the period and C_i the worst-case transmission time of frames sent on stream S_i. The worst-case latency R_i of a CAN frame sent on stream S_i is defined by

$$R_i = J_i + q_i + C_i \tag{1}$$

where J_i is the queuing jitter of the frame, i.e., the maximum variation in queuing time relative T_i, inherited from the sender task which queues the frame, and q_i represents the effective queuing time, given by:

$$q_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j + E(q_i + C_i) \tag{2}$$

where the term B_i is the worst-case blocking time of frames sent on S_i, $hp(i)$ is the set of streams with priority higher than S_i, τ_{bit} (the bit-time) caters for the difference in arbitration start times at the different nodes due to propagation delays and protocol tolerances, and $E(q_i + C_i)$ is an error term denoting the time required for error signalling and recovery. The reason for the blocking factor is that transmissions are non-pre-emptive, i.e., after a bus arbitration has started, the frame with the highest priority among competing frames will be transmitted till completion, even if a frame with higher priority gets queued before the transmission is completed. However, in case of errors a frame can be interrupted/pre-empted during transmission, requiring a complete retransmission of the entire frame. The extra cost for this is catered for in the error term E above.

4. Effects of phasings

Phasing is a result of varying response times for message producers. When performing analysis of a system where message producing tasks are taking some time, R, before sending a message, this time R is calculated for a worst case behaviour. Whether this time will vary or not is not normally catered for. Sources that cause this time R to vary is for example pre-emption caused by other tasks with higher priority executing on the same CPU. Another source for not fixed R can be varying execution times. Even though the code executed before sending the message perhaps is fixed, and thus the time taken to execute this code should be fixed, the assumption fails in reality since the execution time varies as a result of the hardware, e.g., caches, shared resources and so on. All of this will give us a response time, R, before the message is sent as can be seen in Figure 1.

In this paper we will focus on the phasings caused by varying execution times of the message producers. This since, traditionally, the use of worst-case execution times reduces the number of possible queuing times for the produced messages. Thus, without taking execution-time variations into account, we believe that we loose valuable information for making decisions and conclusions.

Instead of using worst-case execution times for the message producers as a basis for when messages are sent, we would rather use some distribution. The knowledge of the system is then extracted using simulation of the system.

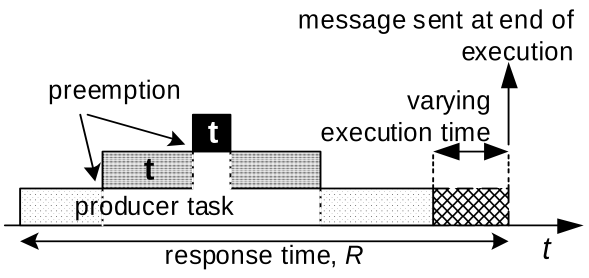

Figure 1. Phasing

4.1 Simulation issues

When simulating a distributed system using a shared communication link, the assumption of fixed execution times is often made, and when performing the simulations, the result can often be a bit optimistic for high priority messages and pessimistic for low priority messages. This since in the simulation messages will be sent on even multiples of their producers periods and thus they will often be treated as if they are sent on exactly the same time. This can, when 2 or more producer has the same period and execution time (in a non-pre-emptive system), result in that the message with the higher priority will be the first message to be sent, as (a) in Figure 2, but in reality, where the execution times of the producers are not fixed, maybe the message with the lower priority is sent slightly before the message with higher priority which will cause the message with the higher priority to be blocked, as (b) in Figure 2. This blocking effect is respected in the analysis (1) of course, but there it is always included, which is pessimistic. In the simulation though, it is important to know of this effect, which, if not considered may exclude the blocking effect.

By considering phasings when performing simulation of a distributed system give us worst-case figures more close to the analytical ones, which is correct. Thus the distributions of the figures will be more close to the distributions that could be expected from a real system. By looking into the response time distributions, we

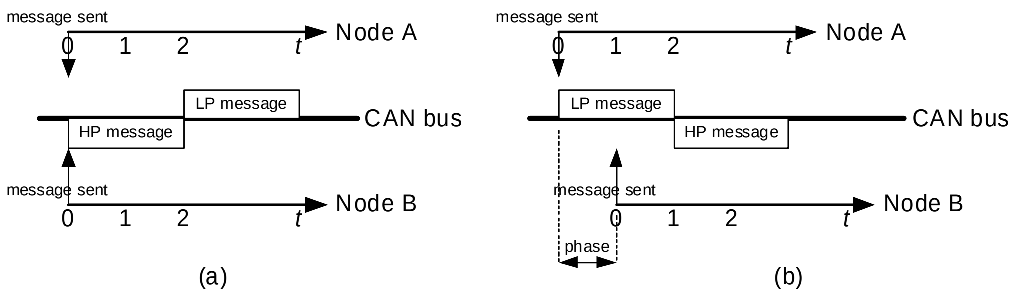

Figure 2. Blocking effects cased by excluding phasing

can see that we have a more natural behaviour compared to the distribution collected when we do not consider phasing.

5. Case-study

Here we will investigate the effects of phasings by simulating a small automotive distributed system: the ABS system of a car, shown in Figure 3. In this ABS system, 4 of the 5 nodes are monitoring nodes, which execute the same code periodically, and thus the time spent executing before the message is sent is traditionally assumed to be fixed, i.e., variations in execution time phasings are not considered. So, the assumption for this system is that messages will be sent on multiples of the producers periods, with some fixed offset caused by the execution time spent before sending the message. This offset will be fixed when not taking the effect of varying execution time phasings into account, but when phasings are considered, this time will vary.

First we will perform simulation of this small subsystem without taking the effect of phasings into account. After this, we will simulate the same subsystem again but now considering phasings between the nodes in the system. The system that we will simulate consists of 4 monitoring nodes and 1 master node, all nodes communicating using a CAN bus, as can be seen in Figure 3. The worst-case response time is also calculated for each message using (1) and presented along with the other system parameters ($P =$ priority of the message, $T =$ period of the producer, $D =$ deadline for the message and C is the transmission time for the message) in Table 1. The monitoring nodes are periodically sending messages to the decision-making node, the master. What we will measure during the simulation of the system are the response time distributions for all the messages sent from of all 4 monitoring nodes, ABS1-ABS4.

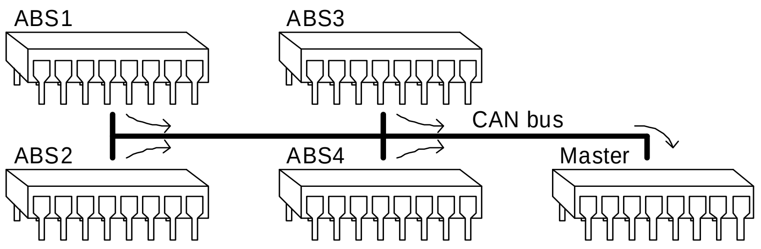

Figure 3. ABS subsystem

5.1. Simulation under worst-case assumptions

The first simulation was performed without considering phasings. We performed exhaustive simulation, where all messages were sent assuming fixed producer-task execution times. All messages were sent at the end of the producer tasks fixed executions, which we mark time 0, and the simulations were performed throughout the least common multiple, LCM, of the participating messages periods, in this case 4 time units. All messages were monitored and their response times were collected. The result of the simulation can be found in Table 2.

Msg ID	P	T_i	D_i	C_i	Sender	R_{calc}
ABS-1	1	4	4	0.54	ABS1	1.08
ABS-2	2	4	4	0.54	ABS2	1.62
ABS-3	3	4	4	0.54	ABS3	2.16
ABS-4	4	4	4	0.54	ABS4	2.16

Table 1. ABS subsystem message set

Msg ID	R_{min}	R_{max}	R_{calc}
ABS-1	0.54	0.54	1.08
ABS-2	1.08	1.08	1.62
ABS-3	1.62	1.62	2.16
ABS-4	2.16	2.16	2.16

Table 2. Simulation results with no phasings

What we can see in these results is that, since all messages are sent at the same time, the message with the highest priority always get served first, giving it an optimal response time of, in this case, 0.54 time units. This is much better that the calculated response time shown in Table 1, and thus optimistic. The same optimism can be found for the other messages, except for the message with the lowest priority.

This simple example clearly shows us the optimism, due to the lack of the blocking effect, presented in Figure 2 (a).

5.2. Simulation with random phasings

In this experiment, the phases were randomly selected using a rectangular distribution, [0, 1]. Here another distribution could be used depending on the knowledge of the system that is to be simulated. When phases were selected, some 70 000 samples were taken and the response times for the 4 different messages are presented in Table 3 and more detailed in Figure 4.

Msg ID	R_{min}	R_{max}	R_{calc}
ABS-1	0.54	1.08	1.08
ABS-2	0.54	1.62	1.62
ABS-3	0.54	2.16	2.16
ABS-4	0.54	2.16	2.16

Table 3. Simulation results with phasings

From the graph we can read that the message with the highest priority, ABS-1, now occasionally is blocked by a lower priority message. This effect was not discovered when simulating the system without phasings. Furthermore, message ABS-2 can be at most blocked by one lower priority message, either ABS-3 or ABS-4, and delayed by message ABS-1, giving us a worst case response time of 1.62. Message ABS-3 can be blocked by message ABS-4 and delayed by both message ABS-1 and ABS-2 giving us a worst-case response time of 2.16. Finally, message ABS-4 can be delayed by all higher priority messages, ABS-1, ABS-2 and ABS-3, but there are no messages with lower priority that can cause blocking. Thus, the worst-case response time is here the same as for message ABS-3, i.e., 2.16. Note that the best case execution time is 0.54 for all messages, so even the message with the lowest priority will sometimes have its message sent directly. However, the important observation here is that the optimism discovered when simulating the system without phasings is now eliminated.

Furthermore, consider the response time distributions in Figure 4, and highlighted in Table 4, where we show the worst-case response times for the lower 50, 75, 80, 90 and 100% of the response times gathered in the simulation. These distributions should be compared with the simulation without considering phasings,

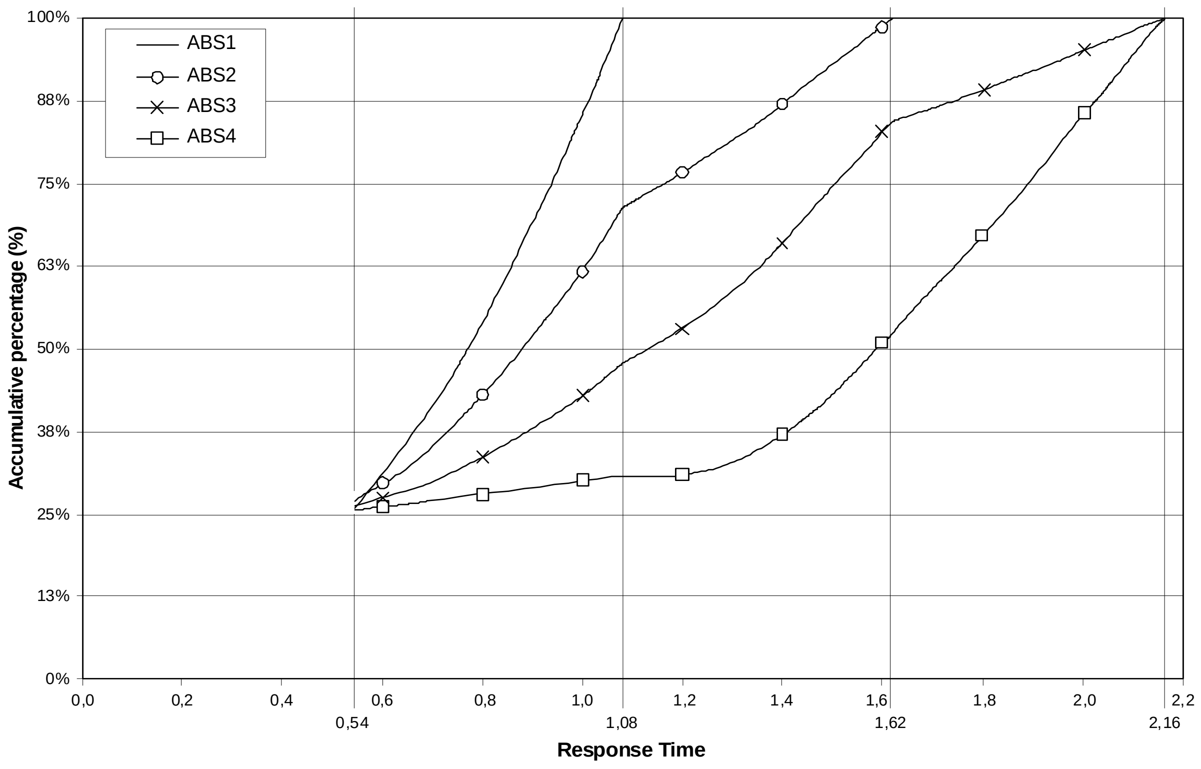

Figure 4. Simulation results when using phasings

where we only have a single fixed worst case response time for each message. Moreover, the response times in that experiment were also optimistic (except the last message, ABS-4, where it only was pessimistic). In the later experiment, when considering phasings, we cover all possible response times (based on our system assumptions), and therefore, by this increased knowledge about the system, we both reduce the pessimism of the lower priority message and we remove the optimism in the first experiment caused by the absence of the blocking factor.

Msg ID	% below response-time				
	50%	75%	80%	90%	100%
ABS-1	0.77	0.94	0.97	1.03	1.08
ABS-2	0.88	1.16	1.27	1.45	1.62
ABS-3	1.13	1.51	1.57	1.83	2.16
ABS-4	1.59	1.89	1.95	2.06	2.16

Table 4. Simulation results with phasings

6. Conclusions

Since the verification of a system or a product typically is based on a model of a system, it is important to reduce the modelled utilisation, i.e., the utilisation given by the model. This can be achieved either by more accurate modelling, or by reducing the actual utilisation of the system. Focusing on message queuing phasings, we have increased the number of possible queuing scenarios and thus increased the knowledge of the system.

We have shown, with a simple case-study, the effects of phasings in a distributed system, where communication is handled using a CAN bus. We have used a case-study, designed to show behaviour caused by message queuings at critical instants in time. We have shown the phasings caused by varying execution times in a small distributed system. We have revealed both optimism and pessimism caused by not taking phasings into account when simulating such a system.

We achieved increased accuracy in the modelling by taking phasings into account, and this give us more accurate response times, as illustrated for our case study in the Table 4.

Future work is to investigate the effects of phasings in a more general perspective. We want to simulate large message sets sent from a large amount of nodes, and we want to perform simulations with specified queuing jitter and period variations. We also want to investigate dependencies between the parameters, e.g., what is the correlation between the actual response times of different frames? The resulting model of response time distributions will provide valuable input to our framework for analysing reliability and timing trade-offs [3]. The ultimate goal is to provide engineers with tools that make it possible to make well founded trade-off decision, leading to both resource lean and sufficiently safe and reliable systems.

References

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

[2] A. Burns. Preemptive Priority Based Scheduling: An Appropriate Engineering Approach. Technical Report YCS 214, University of York, 1993.

[3] H. Hansson, T. Nolte, C. Norström, and S. Punnekkat. Integrating Reliability and Timing Analysis of CAN-based Systems. *IEEE Transaction on Industrial Electronics*. To appear in a special issue on factory communication systems.

[4] H. Hansson, C. Norström, and S. Punnekkat. Integrating Reliability and Timing Analysis of CAN-based Systems. In *Proc. 2000 IEEE International Workshop on Factory Communication Systems (WFCS'2000)*, Porto, Portugal, September 2000. IEEE Industrial Electronics Society.

[5] H. Hansson, C. Norström, and S. Punnekkat. Reliability Modelling of Time-Critical Distributed Systems. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1926 of *Lecture Notes in Computer Science (LNCS)*, 6th International Symposium, FTRTFT 2000, Pune, India, September 2000. Springer-Verlag.

[6] I. S. O. (ISO). Road Vehicles- Interchange of digital information -Controller Area Network (CAN) for high-speed communication. ISO Standard-11898, Nov 1993.

[7] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat. Using bit-stuffing distributions in CAN analysis. *IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01)*.

[8] S. Punnekkat, H. Hansson, and C. Norström. Response Time Analysis under Errors for CAN. In *Proceedings of IEEE Real-Time Technology and Applications Symposium (RTAS 2000)*, pages 258–265. IEEE Computer Society, June 2000.

[9] L. Sha, R. Rajkumar, and J. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[10] K. W. Tindell and A. Burns. Guaranteed message latencies for distributed safety-critical hard real-time control networks. Technical Report YCS229, Dept. of Computer Science, University of York, June 1994.

[11] 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.

[12] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). In *Proceedings 15th IEEE Real-Time Systems Symposium*, pages 259–265. IEEE Computer Society, December 1994.

[13] J. Xu and D. L. Parnas. Priority scheduling versus pre-run-time scheduling. *Real-Time Systems Journal*, 18(1):7–23, January 2000.