Integrating Hardware Limitations in CAN Schedulability Analysis

Dawood Khan, Reinder Bril, Nicolas Navet

Integrating Hardware Limitations in CAN Schedulability Analysis

Dawood A. KHAN
INRIA / INPL
Vandoeuvre-lès-Nancy, France
dawood.khan@loria.fr

Reinder J. BRIL
Eindhoven University of Technology
Eindhoven, Netherlands
r.j.bril@tue.nl

Nicolas NAVET
INRIA / RealTime-at-Work
Vandoeuvre-lès-Nancy, France
nnavet@loria.fr

Abstract

The existing schedulability analysis for the Controller Area Network (CAN) does not take into account that a CAN controller has finite buffer space to store outgoing messages and high priority messages may suffer from priority inversion if the buffers are already occupied by low priority messages. This gives rise to an additional delay for high priority messages, which, if not considered, may result in a deadline violation. In this paper, we explain the cause of this additional delay and extend the existing CAN schedulability analysis to integrate it. Finally, we suggest implementation guidelines that minimizes both the run-time CPU overhead and the additional delay due to priority inversion.

1 Introduction

Context of the study. CAN (Controller Area Network) was specifically designed for use in the automotive domain and has become for more than 10 years a de-facto standard. For instance, high-end cars these days have about 70 CAN controllers. CAN has been extensively used in other areas as well because of its interesting real-time properties and low-cost. Whatever the domain, existing schedulability analysis of real-time applications distributed over CAN assume that:

1. If a CAN node has to send out a stream of messages having the highest priority on the bus, it should be able to do so without releasing the bus between two consecutive messages despite the arbitration process that takes place at the end of each transmission,

2. If on a CAN node more than one message is ready to be sent, the highest priority message will be sent first. This means that the internal organization of the CAN node is such that it is possible.

These assumptions put some constraints on the architecture of the CAN controllers and on the whole protocol stack. Sometimes, because of the CAN controller or protocol layers, priority inversion among messages do occur. These priority inversions induce additional delays to the Worst Case Response Time (WCRT) of the messages, which is being analyzed and integrated into the existing schedulability analysis in this paper.

Limits of existing CAN schedulability analysis. Timing analysis of CAN has been developed over the years [1, 2] but, as pointed out in [3], they usually overlook some of the characteristics of the hardware and the software. In the existing analysis, CAN is modeled as an infinite priority queue in which each node is inserting its messages according to the priority. Then, it is considered that the highest priority message in the queue wins the arbitration. However, this is not always the case in practice. For instance, the CAN controllers have a limited number of transmission buffers and the higher priority messages which get released may get blocked due to non-availability of transmission buffers in a CAN controller. The timing analysis of CAN also ignores the fact that it takes some time to copy messages from higher layers of the protocol stack to the CAN controller transmission buffers and due to this delay, a high-priority message released by the higher protocol layer might miss the start of the arbitration and hence suffer from priority inversion. The existing published analysis also overlook the fact that the queues to hold messages in a protocol layer may be implemented as FIFO and higher priority messages blocked by the lower priority messages in a FIFO will suffer from a priority inversion[1]. In addition, some CAN controllers or drivers do not allow transmission requests to be canceled, which might induce priority inversion. All these factors not covered by the existing analysis have been precisely classified and explained in [3]. However, to the best of our knowledge most of these distortions to the ideal case are not integrated into any response time analysis.

[1] At least one commercial tool, namely NETCAR-Analyzer from RTaW (see `http://www.realtimeatwork.com/?page_id=396`), addresses the FIFO case.

Contributions of the paper. The effects of a limited number of transmission buffers have been identified in [4], [5] and [6]. In [5] the author gives the analysis for the case when it is not possible to cancel transmission and in [6] the authors show that at least 3 transmission buffers are needed to avoid priority inversions when the copying time of a message from the queue to the controller is neglected (see section 2). Here, we address the 3 or more buffer case when it is possible to cancel a transmission request and when the copying overhead can take any reasonable value, and we derive a worst-case response time analysis that integrates it. Besides, we provide guidelines for an optimized CAN driver implementation. The case addressed here is meaningful because in practice most CAN controllers have more than 3 buffers and possess the ability to cancel a transmission request.

2 Priority inversion

When all the transmission buffers in a CAN controller are filled and a message is released; assuming the newly released message is of lower priority than the messages in transmission buffer, then the newly released message waits in the priority queue for the availability of one transmission buffer. However, if this newly released message is of higher priority than those in transmission buffers then - to respect the highest priority first (HPF) principle underlying CAN - it should be swapped with the lowest priority message in transmission buffers that is not undergoing transmission. Moreover, if the bus arbitration starts anytime during the swapping process (*i.e.*, lower priority message put back in the queue, higher priority message copied into the freed buffer), it may happen that a lower priority message, be it on the same station or elsewhere on the network, win the arbitration. The higher priority message will then suffer from priority inversion and its Worst-Case Response Time (WCRT) might be larger than what is given by the current CAN schedulability analysis. This additional delay is caused by the fact that a buffer place needs to be made free for this newly arriving message of high priority.This increase in WCRT is modeled by a factor called the Additional Delay (AD) in the rest of the paper. An example of how AD occurs is shown in figure 1.

3 System model

We assume a set $\mathcal{M}$ of m messages[2] μ_1, μ_2, ..., μ_m, where $m \in \mathbb{N}$. Each message μ_i is characterized by a *period* $T_i \in \mathbb{R}^+$, an activation jitter J_i, a worst-case transmission time $C_i \in \mathbb{R}^+$ an additional delay AD_i, and a (*relative) deadline* $D_i \in \mathbb{R}^+$, where $D_i \leq T_i$. Besides, one defines the maximum copying time CT_i for μ_i as the maximum between the time needed to copy the message

[2]Here the term message denotes the payload of a frame (not an individual signal). In Autosar terminology it would be termed L-PDU.

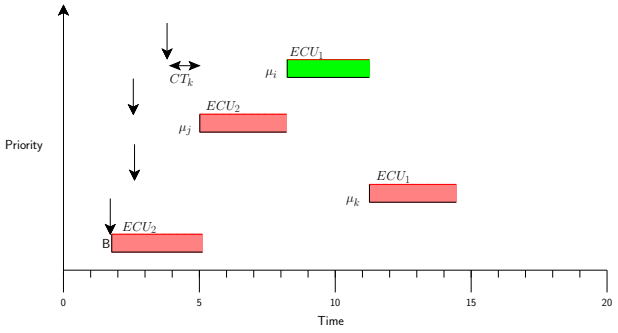

Figure 1. Message μ_i is released while a lower priority frame is being sent (blocking delay B). The transmission buffers on ECU1 are full, the device driver then aborts lower priority message μ_k and copies it into queue taking time CT_k.Then μ_i is copied into the freed transmission buffer taking time CT_i. However, while μ_i is being copied the arbitration is lost to message μ_j and μ_i suffers an additional delay of $AD = CT_k + C_j - B$ as compared to initial B.It should be pointed out that this additional delay of μ_i appears as an additional jitter to lower priority message μ_k.

from the queue to the transmission buffer and the time to copy from the buffer to the queue[3]. Here, we make the reasonable assumption that the copying time is less than the transmission time of the smallest frame. For notational convenience, we assume that the messages are given in order of decreasing priority, i.e. μ_1 has highest priority and μ_m has lowest priority. Moreover, we assume a set $\mathcal{C}$ of n CAN controllers CC_1, CC_2, ..., CC_n, where $n \in \mathbb{N}$. Each CAN controller CC_c has $k_c \in \mathbb{N}$ transmission buffers[4]. Furthermore, we are assuming that multiple transmission buffers on CAN controllers are not occupied by messages of the same priority. A total function $CC : \mathcal{M} \rightarrow \mathcal{C}$ defines which message is sent by which CAN controller. The set of messages M_c sent by controller CC_c is defined as

$$M_c = \{\mu \in \mathcal{M} | CC(\mu) = CC_c\}. \quad (1)$$

The additional delay AD_i of a message μ_i due to priority inversion appears as an additional jitter to lower priority messages, in addition to original queuing jitter J_i.The time from the release of μ_i to the arrival of μ_i in the queue is classically defined as the jitter of μ_i. After the jitter J_i the message μ_i is considered to be able to participate in arbitration, which is false in case of priority inversion. The controllers $\{CC_i \, s.t. \, i \neq c\}$ can observe the interference

[3]Both delays could be distinguished but in practice we expect them to be very similar.

[4]The exact hardware architecture may vary as for instance we may have messages being mapped to a specific buffer on CAN controllers with multiple transmission buffers.

of message μ_i when it is able to participate in the arbitration after priority inversion (after AD_i) and thus AD_i acts as an extension to the original jitter.Therefore, the total jitter for μ_i seen by lower priority messages is:

$$\hat{J}_i = J_i + AD_i \tag{2}$$

The Worst Case Response Time (WCRT) of a message is defined as the maximum possible time taken by the message to reach the destination CAN controller from the time of invocation at the sending task. A message μ_i is said to be schedulable if and only if its WCRT R_i is less than or equal to its relative deadline D_i and system is schedulable if and only if all the messages on the CAN network are schedulable.

4 Response time analysis when transmission requests can be aborted

This section provides the method to compute the worst-case response time of messages on the CAN network. The computed values are then used to check the schedulability of the system by comparing the WCRTs against the deadlines. The analysis given in this paper provides a simple and non-necessary schedulability condition directly inspired from [2]. It assumes no errors on the bus but they can be included as classically done in [1]. Following the analysis given in [1, 2] the worst-case response time can be described as a composition of three elements:

1. the queuing jitter J_i, the longest time it takes to queue the message starting from initiating event,
2. the queuing delay w_i, the longest time for which a message can remain in the driver queue or transmission buffers before successful transmission,
3. the worst-case transmission time C_i, the longest time a message can take to be transmitted.

A bound on the worst-case response time of a message μ_i is therefore given as:

$$R_i = J_i + w_i + C_i \tag{3}$$

The queuing delay w_i is composed as follows:

1. blocking delay which is the delay due a lower priority frame that has started to be transmitted before μ_i can participate to the arbitration, plus possibly the time needed to free a buffer on the ECU of μ_i (see §4.2),
2. the delay due to interference of higher priority messages which may win the arbitration and transmit one or several times before μ_i.

When computing bound on the response times, we can distinguish two cases i) messages which are safe from priority inversion ii) messages which suffer from priority inversion and will be swapped with the lowest priority message in transmission buffers not in transmission.

4.1 Case 1: messages safe from priority inversion

It should be noticed that the higher priority messages on each CAN controller CC_l are more susceptible to priority inversion as compared to lower priority messages on the same CAN controller. Indeed, the k_l lowest priority messages on CC_l will not suffer from any priority inversion as all the k_l transmission buffers cannot be occupied by lower priority messages, thus these messages are not suffering from any *additional delay*. For these messages, the worst-case queuing delay, using the model given by [2], will be given by:

$$w_i^{n+1} = \max(B_i, C_i) + CT_i + \sum_{\forall k \in hp(\mu_i)} \lceil \frac{\hat{J}_k + w_i^n + \tau_{bit}}{T_k} \rceil C_k \tag{4}$$

where $\hat{J}_k$ is computed using (2) and B_i is the maximum blocking time due to lower priority messages which occurs when the lower priority message of largest size has just started to be transmitted when μ_i arrives, i.e.

$$B_i = \max_{k \in lp(\mu_i)} \{C_k\} \tag{5}$$

A suitable starting value for the recurrence relation given above is $w_i^0 = C_i$. This relationship keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. And if the message is schedulable its WCRT will be given by (3).

4.2 Case 2: messages undergoing priority inversion

Messages not belonging to the k_l lowest priority messages can suffer from priority inversions when all the k_l transmission buffers are filled up with lower priority messages. We consider here the case where the communication driver will abort a transmission request whenever a message that possesses a higher priority than those already in the transmission buffers arrives, let's say μ_i. Specifically, the CAN driver will abort the lowest priority message on CC_c *not currently under transmission* and start copying μ_i in place. The swapping of μ_i will induce some delay and if arbitration starts during the swapping process a lower priority message than μ_i may win arbitration and starts to transmit. This may introduce an additional delay AD_i for μ_i which is equivalent to the difference between the transmission time of the message which won arbitration and the original blocking delay B_i, plus the time needed to copy a message from the communication buffer to the queue. The worst-case AD_i is obtained by taking the maximum of the worst-case transmission times for all values of k such that $i < k \leq j$ where μ_j is the highest priority message of the lowest k_l priority messages on CC_l:

$$AD_i = \max\left(0, \max_{\{\forall k \in M_C | k > i\}} (CT_k) + \max_{i < k \leq j} (C_k) - B_i\right) \tag{6}$$

where CT_k is the copy time of the message which is replaced by μ_i. Then, the worst-case queuing delay for message μ_i is given by:

$$w_i^{n+1} = \max(\hat{B}_i, C_i) + CT_i + \sum_{\forall j \in hp(\mu_i)} \lceil \frac{\hat{J}_j + w_i^n + \tau_{bit}}{T_j} \rceil C_j \quad (7)$$

where $\hat{J}_j$ is given by (2) and $\hat{B}_i$ is given by $B_i + AD_i$. A suitable starting value for the recurrence relation give above is $w_i^0 = C_i$. This relationship keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. And if the message is schedulable its WCRT will be given by (3).

5 Optimized implementation and case-study

If we accept the overhead of keeping a copy of the messages currently in the transmission buffers in the priority queue, we can suppress an extra copy time and remove the quantity $\max_{\{\forall k \in M_C | k > i\}} CT_k$ in (6). This can be done by maintaining an extra status field along with the priority queue. For instance, for the messages in the transmission buffers this field could be set to one and for the messages in priority queue but not in any transmission buffer this field could be set to zero. Upon the successful transmission of a message its corresponding copy along with its status field will be removed from the priority queue.

Upon a full transmission buffers, for any new message with priority greater than any message in the transmission buffers, it will be first put in the priority queue then the status field of message in transmission buffers with lowest priority and not transmitting will be set to zero. Then the message will over-write the message in transmission buffer whose field was just set to zero and finally for the message which replaced the message in the transmission buffer, the status field is set to one. This procedure will remove the need for swapping which takes more time as compared to simple overwrite and thus chances of priority inversion are reduced. However, the downside of this is that we have to re-arrange the priority queue not only each time a message becomes available but also each time a message is successfully sent by the station (upon the acknowledgment).

We illustrate the analysis on an typical 125Kbit/s automotive body network. To generate a realistic test network we used *Netcarbench* [7]. The generated periodic message sets under study consists of 105 CAN messages mapped over 17 ECUs with deadlines equal to periods and data payload ranging from 1 to 8 bytes. The total periodic load is equal to 42.04%.

Figure 2 shows the worst-case response times of the CAN messages with and without priority inversion. We observe the impact on the WCRT of messages when priority inversion is taken into account. For instance in figure 2, the WCRT for the message with id 101 raises from 100.8ms without priority inversion to 120ms (*i.e.* 19% increase).

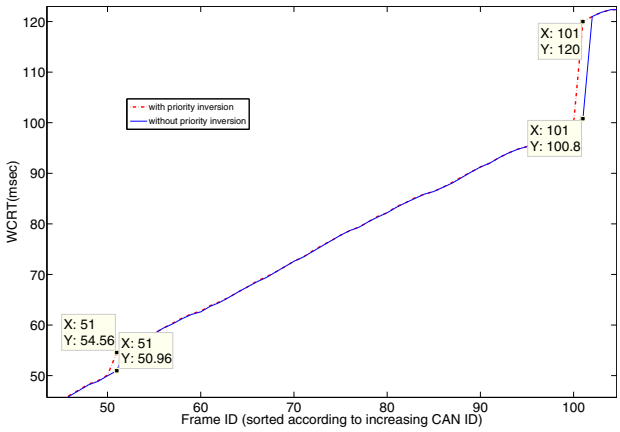

Figure 2. Worst-case response time with and without taking into account priority inversion. Only frames starting from ID 40 are shown.

6 Conclusion

The aim of the paper is to understand and analyse the consequences of architectural limitations in CAN. Here, we derive a more realistic response time analysis in the typical case where controllers have three or more transmission buffers and the ability to cancel transmission requests. Our future work is to address the case where the CAN controller or the CAN driver does not support an abort mechanism, which greatly increases the distortion with regard to the ideal CAN case.

References

[1] K. Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network (CAN) message response times", *Control Engineering Practice*, vol. 3, no. 8, pp. 1163 – 1169, 1995.

[2] R. Davis, A. Burn, R. Bril, and J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised", *Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[3] M. D. Natale, "Understanding and using the Controller Area Network", Handout of a lecture at U.C. Berkeley available at `http://inst.eecs.berkeley.edu/~ee249/fa08/`, October 2008.

[4] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communications: Controller Area Network (CAN)", in *Real-Time Systems Symposium*, Dec 1994, pp. 259–263.

[5] M. D. Natale, "Evaluating message transmission times in Controller Area Networks without buffer preemption", in *8th Brazilian Workshop on Real-Time Systems*, 2006.

[6] A. Meschi, M. D. Natale, and M. Spuri, "Priority Inversion at the Network Adapter when Scheduling Messages with Earliest Deadline Techniques", in *8th Euromicro Workshop on Real-Time Systems*, pp. 243–248. IEEE Computer Society, June 1996.

[7] C. Braun, L. Havet, and N. Navet, "NETCARBENCH: a benchmark for techniques and tools used in the design of automotive communication systems", in *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, 2007, pp. 321–328.