Extending Response-Time Analysis of Mixed Messages in CAN with Controllers Implementing Non-Abortable Transmit Buffers

Saad Mubeen*, Jukka Mäki-Turja*† and Mikael Sjödin*
*Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden
†Arcticus Systems, Järfälla, Sweden
{saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se

Abstract

The existing response-time analysis for messages in Controller Area Network (CAN) with controllers implementing non-abortable transmit buffers does not support mixed messages that are implemented by several high-level protocols used in the automotive industry. We present the work in progress on the extension of the existing analysis for mixed messages. The extended analysis will be applicable to any high-level protocol for CAN that uses periodic, sporadic and mixed transmission modes and implements non-abortable transmit buffers in CAN controllers.

1 Introduction

Controller Area Network (CAN) [1] is a well-known bus communication protocol for real-time applications in automotive domain. According to CAN in Automation, the estimated number of CAN enabled controllers sold in 2011 were about 850 million and most of them were used for automotive applications. CAN is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 Mbits/sec. There are several high-level protocols for CAN that are developed for various industrial applications such as CAN Application Layer, CANopen, Hägglunds Controller Area Network (HCAN), CAN for Military Land Systems domain (MilCAN).

1.1 Background and related work

Tindell et al. [2] developed the Response Time Analysis (RTA) for CAN by adapting the theory of fixed priority preemptive scheduling for uniprocessor systems. Later on, Davis et al. [3] refuted, revisited and revised the analysis developed by Tindell et al. The queueing polices implemented by the CAN device drivers and communications stacks, internal organization and hardware limitations of CAN controllers may have significant impact on the timing behavior of CAN messages [4]. A few examples of such limitations are controllers implementing FIFO and work-conserving queues [4, 5], limited number of transmit buffers [6, 7, 8], copying delays in transmit buffers [6, 8], transmit buffers supporting abort requests [7], the device drivers lacking abort request mechanisms in transmit buffers [4, 6, 7, 8] and protocol stack prohibiting transmission abort requests in some configurations as in the case of AUTOSAR [9].

The research community has targeted these issues and accordingly extended RTA for CAN [2]. RTA in [2, 3] is extended in [5] which is applicable to CAN network where some nodes implement priority queues and some implement FIFO queues. This analysis was further extended for messages with arbitrary deadlines in FIFO and work-conserving queues [4]. However, the analysis in [2, 3] assumes that CAN controllers have very large number of transmit buffers. However, most CAN controllers have small number of transmit buffers [6, 4]. If all such buffers are occupied by lower priority messages, a higher priority message released in the same controller may suffer from priority inversion (it will be discussed in Section 3) [2, 7, 8]. If the controller supports transmission abort requests then the lowest priority message in the transmit buffer (not under transmission) is swapped with the higher priority message from the message queue at the cost of additional delay that was integrated by Khan et al. [7] with the existing analysis [3]. In the case of non-abortable transmit buffers, RTA of CAN messages is extended in [6, 8]. However, none of the above analyses support RTA of mixed messages (see Section 2) in CAN.

1.2 Previous work

In [10], we extended the existing analysis for CAN [2, 3] to support mixed messages. This analysis has been implemented in the existing industrial tool suite, i.e., Rubus-ICE [11, 12, 13]. In [14, 15], we further extended the previous analysis [10] by integrating it with the analysis in [5] to support response-time computation of mixed messages in CAN with priority- and FIFO-queued nodes. In [16], Mubeen et al. developed offset-set aware analysis for mixed messages in CAN. In [17], Mubeen et al. extended the existing RTA for mixed messages in CAN with controllers supporting transmission abort requests in transmit buffers. However, none of the above RTA for mixed messages support non-abortable transmit buffers in CAN controllers.

1.3 Motivation

The motivation for this work comes from the need to conduct an automotive-application case study that involves the modeling and analysis of a distributed embedded system employing CAN for network communication. The ECUs (Electronic Control Units) that are connected to a CAN bus communicate by means of periodic, sporadic and mixed messages. Moreover, the ECUs are heterogeneous, i.e., some controllers implement priority queues, some implement FIFO queues, some support transmission abort requests and some implement non-abortable transmit buffers. The problem is that the existing RTA for mixed messages in CAN does not support the analysis of systems where ECUs implement non-abortable transmit buffers.

1.4 Paper contribution

We present the work in progress on the extension of the existing analysis for mixed messages in CAN [10] by integrating it with the analysis in [6]. Mixed messages represent a common message transmission pattern which is implemented by some high-level protocols used in the automotive industry today. Further, the existing analysis in [6] places a restriction on message deadline, i.e., the deadline should be less than or equal to the period of the message. On the other hand, we assume arbitrary deadlines, i.e., the deadline of a message can be higher than its period. The extended analysis will be generally applicable to any high-level protocol for CAN that uses periodic, sporadic, and mixed transmission of messages and supports CAN controllers that implement non-abortable transmit buffers.

2 Implementation of mixed messages by high-level protocols

A mixed message can be queued for transmission periodically as well as sporadically, i.e., it is simultaneously time and event triggered. We identified three different methods for mixed message implementation by high-level protocols, i.e., CANopen [18], AUTOSAR [19] and HCAN [20] in [14]. Due to space limitation, we only discuss the implementation of a mixed message in HCAN protocol in detail and compare it with the rest of the implementations.

A mixed message defined by HCAN protocol contains periodic and sporadic signals. It is queued for transmission not only periodically, but also as soon as an event occurs that changes the value of one or more event signals, provided Minimum Update Time (MUT) between the queueing of two successive sporadic instances of the mixed message has elapsed. Hence, the transmission of a mixed message due to arrival of events is constrained by MUT. The transmission pattern of a mixed message implemented by HCAN protocol is depicted in Figure 1.

Message 1 is queued because of partially periodic nature of a mixed message. As soon as event A arrives, message 2 is queued for transmission and MUT timer is started. When event B arrives it is not queued immediately because MUT is not expired yet. As soon as MUT expires, message 3 is queued. Message 3 contains the signal changes that correspond to event B. Similarly, a message is not immediately queued when event C arrives because MUT is not expired. Message 4 is queued because of the periodicity. Although, MUT was not yet expired, the event signal corresponding to event C was packed in message 4 and queued as part of the periodic message. Hence, there is no need to queue an additional sporadic message when MUT expires. This indicates that the periodic transmission of a mixed message cannot be interfered by the sporadic transmission (a unique property of HCAN protocol). When event D arrives, a sporadic instance of the mixed message is immediately queued as message 5 because MUT has already expired. Message 6 is queued due to periodicity.

It can be seen from the queuing of instances 4 and 6 of the mixed message in Figure 1 that the periodic transmission is independent of the sporadic transmission. A mixed message can be queued for transmission even if MUT is not expired. This shows that the worst-case periodicity of a mixed message implemented by HCAN is neither bounded by period nor by MUT. Since, the existing analysis for CAN with controllers implementing non-abortable transmit buffers [6] is based on the assumption that the worst-case periodicity of a message is either bounded by its period or MUT, it cannot be used for mixed messages.

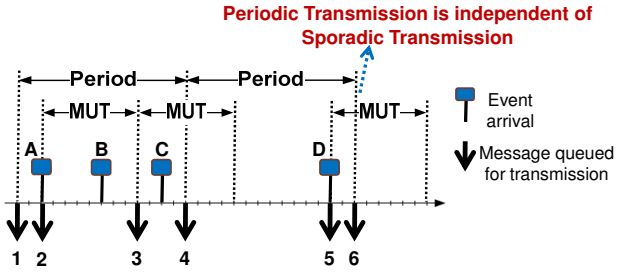

Figure 1. Mixed transmission pattern in HCAN

On the other hand, there exists a dependency relation between the periodic and sporadic transmissions of a mixed message implemented by CANopen and AUTOSAR. If the same mixed message is implemented by CANopen and AUTOSAR then the periodic transmissions of the mixed message corresponding to 4 and 6 in Figure 1 will be delayed until the expiry of MUT timer. In CANopen, the periodic timer is reset with every periodic or sporadic transmission. Whereas in AUTOSAR, the periodic transmission is delayed until the expiry of sporadic timer. Hence, the worst-case periodicity of a mixed message in CANopen and AUTOSAR can never be higher than the sporadic timer (called Inhibit Timer in CANopen and Minimum Delay Timer in AUTOSAR). Intuitively, the mixed message in CANopen and AUTOSAR can be treated as a special type of sporadic message. Therefore, the existing analysis [6] holds good for the implementations of a mixed message in CANopen and AUTOSAR.

3 System scheduling model

The system scheduling model is inspired by the model developed by Tindell et al. [2]. It combines the system model of RTA of CAN for mixed messages [10] with the scheduling model in [7]. The system consists of a number of CAN controllers (nodes), i.e., $CC_1, CC_2, ...CC_n$ which are connected to a single CAN network. The nodes implement priority-ordered queues. The total number of messages in the system are defined in a set $\aleph$. Let a set $\aleph_c$ defines the set of messages sent by a CAN controller CC_c. We assume that each controller has a finite number of transmit buffers. Let K_c denote the number of transmit buffers in a CAN controller CC_c. Each CAN message m has an ID_m which is a unique identifier. P_m denotes a unique priority of m. We assume that the priority of a message is equal to its ID. The priority of m is considered higher than the priority of another message n if $P_m < P_n$. Let the sets $hp(m)$, $lp(m)$, and $hep(m)$ contain the messages with priorities higher, lower, and equal and higher than m respectively. $\xi(m)$ denotes the transmission type that specifies whether a message is periodic (P), sporadic (S) or mixed (M). Formally the domain of $\xi(m)$ is defined as:

$$\xi(m) \in [P, \quad S, \quad M]$$

Each message has a transmission time (C_m) and queueing jitter (J_m). J_m is inherited as the difference between the worst- and best-case response times of the queueing task. Each message can carry a data payload (ranges from 0 to 8 bytes) denoted by s_m. In the case of periodic transmission, each message has a period denoted by T_m. Whereas in the case of sporadic transmission, each message has a MUT_m that refers to the minimum time that should elapse between the transmission of any two sporadic messages. Each message has a blocking time (B_m) which refers to

the largest amount of time m can be blocked by any lower priority message. R_m denotes the Worst Case Response Time (WCRT) of m and is defined as the longest time between the queueing of the message (on the sending node) and the delivery of the message to the destination buffer.

We duplicate a message when its transmission type is mixed and treat it as two separate messages, i.e., periodic and sporadic. All attributes of these duplicates are the same except the periodic copy inherits T_m while the sporadic copy inherits MUT_m. A system is considered schedulable if all of its messages are schedulable. A message m is deemed schedulable if its R_m is less than or equal to its D_m. We assume arbitrary deadlines, i.e., the deadline of a message can be greater than, equal to or less than its period or MUT. We further assume that CAN controllers are capable of buffering more than one instance of a message.

Additional Delay due to Priority Inversion. When CAN controllers do not support transmission abort requests, a higher priority message may suffer from priority inversion and this, in turn, adds extra delay to its response time [6]. Consider an example of three controllers CC_c, CC_j, CC_k connected to CAN in Figure 2. Let m_1, belonging to CC_c, be the highest priority message in the system. Assume that when m_1 is ready to be queued, all transmit buffers in CC_c are occupied by lower priority messages which can not be aborted. Moreover, m_1 can also be blocked by any message in the set $lp(m)$ (m_5 in this case). Therefore, m_1 has to wait in the priority queue until one of the messages in K_c are transmitted. Let m_4 be the highest priority message in K_c. m_4 can be interfered by higher priority messages belonging to other nodes in the system (m_2 and m_3). Hence, it can be seen in this example that priority inversion takes place because m_1 cannot start its transmission before m_4 finishes its transmission while m_4 has to wait until messages m_2 and m_3 are transmitted. Let the additional delay for m due to priority inversion be denoted by AD_m.

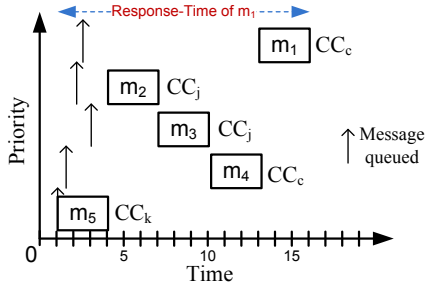

Figure 2. Demonstration of priority inversion

4 Extended analysis

Let m be the message under analysis belonging to node CC_c. We treat m differently if it is periodic, sporadic or mixed. A message may or may not suffer from priority inversion [6]. For example, if K_c is equal to 3 then last three lowest priority messages cannot face priority inversion. We will consider four cases in the extended analysis as follows.

1. Case 1: When m is safe from priority inversion.
   (a) When $\xi(m)$ is periodic or sporadic.
   (b) When $\xi(m)$ is mixed.
2. Case 2: When m is subjected to priority inversion.
   (a) When $\xi(m)$ is periodic or sporadic.
   (b) When $\xi(m)$ is mixed.

Due to lack of space, we only discuss the extended analysis in case 2(b). Since, a mixed message is duplicated, we compute the response time of both the duplicates separately. We denote the periodic and sporadic copies of a mixed message m by m_P and m_E respectively. Let WCRT of m_P and m_E be denoted by R_{m_P} and R_{m_E} respectively. WCRT of m is equal to the largest value between R_{m_P} and R_{m_E} as follows.

$$R_m = max(R_{m_P}, \quad R_{m_E}) \tag{1}$$

Let us denote the total number of instances of m_P and m_E arriving in the priority level-m busy period by Q_{m_P} and Q_{m_E} respectively. Assume that the index variable for message instances of m_P and m_E is denoted by q_{m_P} and q_{m_E} respectively. The range of q_{m_P} and q_{m_E} is given by:

$$0 \leq q_{m_P} \leq (Q_{m_P} - 1) \;\; ; \;\; 0 \leq q_{m_E} \leq (Q_{m_E} - 1) \tag{2}$$

WCRTs of m_P and m_E are equal to the largest value among their respective response times of all instances arriving in the busy period as shown below.

$$R_{m_P} = max(R_{m_P}(q_{m_P})) \;\; ; \;\; R_{m_E} = max(R_{m_E}(q_{m_E})) \tag{3}$$

Due to space limitation, we only discuss the computation of WCRT of each instance of m_P by adapting the existing analysis of mixed messages [10]. WCRT of each instance of m_E can be computed in a similar fashion.

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P}T_m + C_m \tag{4}$$

C_m in (4) is calculated according to the existing analysis [3]. Although, both the duplicates of m inherit same J_m and C_m from it, they experience different amount of worst-case queueing delay caused by other messages.

Worst-case queueing delay. The worst-case queueing delay experienced by m_P, denoted by ω_{m_P} in (4) consists of three factors.

1. The blocking delay which is the maximum value between blocking time (B_m) and additional delay (AD_m) which were discussed in Section 3.
2. Interference from higher priority messages.
3. Self interference, i.e., m_P can be interfered by m_E and vice versa.

ω_{m_P} can be computed by integrating the existing analysis for mixed messages [10] with [6].

$$\omega^{n+1}_{m_P}(q_{m_P}) = \hat{\mathrm{B}}_m + q_{m_P}C_m + \sum_{\forall k \in hp(m)} I_{k_P}C_k + Q^P_{m_E}C_m \tag{5}$$

(C_m+AD_m) can be selected as the initial value of the queueing delay [6]. I_{k_P} is given by (6).

$$I_{k_P} = \begin{cases} \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + \hat{\mathrm{J}}_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{P} \\ \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + \hat{\mathrm{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{S} \\ \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + \hat{\mathrm{J}}_k + \tau_{bit}}{T_k} \right\rceil + \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + \hat{\mathrm{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{M} \end{cases} \tag{6}$$

It is evident from (6) that m_P receives double interference from every higher priority mixed message. Note that the jitter J_k is replaced with increased jitter $\hat{\mathrm{J}}_k$ compared to the existing analysis [10]. This is because the *Additional Jitter* (AJ) received by the higher priority message k due to priority inversion will contribute to the response time of m as an additional jitter of k apart from J_k as shown below.

$$\hat{\mathrm{J}}_k = J_k + AJ_k \tag{7}$$

$\hat{\mathrm{B}}_m$ in (5) is adapted from [6]. m can be blocked by any message in the set $lp(m)$, previous instance of m (push-through blocking [3]) or due to additional blocking because of priority inversion. Hence, it is the maximum value among B_m, C_m and AD_m.

$$\hat{\mathrm{B}}_m = \max(B_m, C_m, AD_m); \quad where, \quad B_m = \max_{\forall k \in lp(m)}(C_k) \tag{8}$$

The computation of additional jitter in (7) and additional delay in (8) for a mixed message m is the work in progress.

Effect of self interference. The effect of self interference can be seen in the last term of (5) . $Q^P_{m_E}$ denotes the total number of instances of m_E that are queued ahead of $q^{th}_{m_P}$ instance of m_P. We reuse $Q^P_{m_E}$ that we derived in [10] with a slight modification (i.e., J_m is replaced with $\hat{\mathrm{J}}_m$).

$$Q^P_{m_E} = \left\lceil \frac{q_{m_P} T_m + \hat{\mathrm{J}}_m}{MUT_m} \right\rceil \tag{9}$$

Length of the busy period. The length of priority level-m busy period, denoted by t_m, can be computed using [10].

$$t^{n+1}_m = \hat{\mathrm{B}}_m + \sum_{\forall k \in hep(m)} I'_k C_k \tag{10}$$

I'_k in (10) is given by the following relation. Note that the contribution of both the duplicates of every mixed message k in a set $hep(m)$ is taken into account.

$$I'_k = \begin{cases} \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{T_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{P} \\ \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{S} \\ \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{T_k} \right\rceil + \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{M} \end{cases} \tag{11}$$

Since the duplicates of a mixed message inherit the same priority from it, the contribution of delay from the duplicate is also covered by using $hep(m)$ in (10). C_m can be used as an initial value of t^n_m in (10). The number of instances of m_P that become ready for transmission just before the end of busy period, i.e., Q_{m_P} can be computed as follows.

$$Q_{m_P} = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{12}$$

5 Summary

The existing response-time analysis for mixed messages in CAN assumes that CAN controllers have large number of transmit buffers. However, some CAN controllers have small number of transmit buffers. If transmission abort requests are not supported by CAN controller device drivers or protocol stack then a higher priority message may undergo priority inversion if all transmit buffers are occupied by lower priority messages. Due to these hardware and software limitations, an additional delay is contributed to the response time of messages.

The existing analysis of CAN supporting non-abortable transmit buffers does not support mixed messages which are implemented by several high-level protocols for CAN used in the industry today. We presented the work in progress on the extension of the existing analysis to support mixed messages in CAN network where CAN controllers do not support transmission abort requests in the transmit buffers. Once the analysis is fully developed, we will combine it with the analysis of mixed messages in CAN supporting transmission abort requests [17] in the longer version of this paper. We plan to implement the extended analysis in the existing industrial tool suite (Rubus-ICE) and conduct the industrial case study (discussed in Section 1.2).

References

[1] Robert Bosch GmbH, "CAN Specification Version 2.0," postfach 30 02 40, D-70442 Stuttgart, 1991.

[2] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communications: controller area network (CAN)," in *Real-Time Systems Symposium (RTSS) 1994*, pp. 259 –263.

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

[4] R. Davis and N. Navet, "Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-Conserving Queues," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, may 2012, pp. 33 –42.

[5] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller Area Network (CAN) Schedulability Analysis with FIFO queues," in *23rd Euromicro Conference on Real-Time Systems*, July 2011.

[6] D. Khan, R. Davis, and N. Navet, "Schedulability analysis of CAN with non-abortable transmission requests," in *16th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, sept. 2011.

[7] D. Khan, R. Bril, and N. Navet, "Integrating hardware limitations in can schedulability analysis," in *8th IEEE International Workshop on Factory Communication Systems (WFCS)*, may 2010, pp. 207 –210.

[8] M. D. Natale, "Evaluating message transmission times in controller area networks without buffer preemption," in *8th Brazilian Workshop on Real-Time Systems*, 2006.

[9] "Transmit Cancelation in AUTOSAR Specification of CAN Driver, Release 4.0, Rev 3, Ver. 4.0. Nov., 2011," http://www.autosar.org/download/R4.0/AUTOSAR_SWS_CANDriver.pdf.

[10] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending schedulability analysis of controller area network (CAN) for mixed (periodic/sporadic) messages," in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2011.

[11] "Arcticus Systems," web page, http://www.arcticus-systems.com.

[12] S. Mubeen, J. Mäki-Turja and M. Sjödin, "Support for holistic response-time analysis in an industrial tool suite: Implementation issues, experiences and a case study," in *19th IEEE Conference on Engineering of Computer Based Systems (ECBS)*, April 2012, pp. 210–221.

[13] S. Mubeen, J. Mäki-Turja, M. Sjödin, and J. Carlson, "Analyzable modeling of legacy communication in component-based distributed embedded systems," in *37th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Sep. 2011, pp. 229–238.

[14] S. Mubeen, J. Mäki-Turja and M. Sjödin, "Response-Time Analysis of Mixed Messages in Controller Area Network with Priority- and FIFO-Queued Nodes," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, May 2012.

[15] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending response-time analysis of controller area network (CAN) with FIFO queues for mixed messages," in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2011, pp. 1–4.

[16] S. Mubeen, J. Mäki-Turja and M. Sjödin, "Worst-case response-time analysis for mixed messages with offsets in controller area network," in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2012.

[17] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Response Time Analysis for Mixed Messages in CAN Supporting Transmission Abort Requests," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2012.

[18] "CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Version 4.02. February 13, 2002."

[19] "AUTOSAR Techincal Overview, Version 2.2.2., Release 3.1, The AUTOSAR Consortium, Aug., 2008," http://autosar.org.

[20] "Hägglunds Controller Area Network (HCAN), Network Implementation Spec." *BAE Systems Hägglunds, Sweden*, April 2009.