Response Time Analysis for Mixed Messages in CAN Supporting Transmission Abort Requests

Saad Mubeen*, Jukka Mäki-Turja*† and Mikael Sjödin*
* Mälardalen Real-Time Research Centre, 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 CAN controllers facilitating transmission abort requests in transmission buffers does not support mixed messages. The existing analysis assumes that a message is queued for transmission either periodically or sporadically. However, a message can also be queued both periodically and sporadically using a mixed transmission mode implemented by several high-level protocols for CAN used in the industry today. We extend the existing analysis for mixed messages in CAN which is generally applicable to any high-level protocol that uses periodic, sporadic and mixed transmission modes and supports transmission abort requests in CAN controllers.**

I. Introduction

Controller Area Network (CAN) [1] is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 mega bits per second. According to CAN in Automation, the estimated number of CAN enabled controllers sold in 2011 are about 850 million. There are several high-level protocols for CAN that are developed for many industrial applications such as CAN Application Layer (CAL), CANopen, Hägglunds Controller Area Network (HCAN), CAN for Military Land Systems domain (MilCAN).

A. Motivation and Related Work

The schedulability analysis of CAN was developed by Tindell et al. [2] by adapting the theory of fixed priority preemptive scheduling for uniprocessor systems. This analysis has served as the basis for many research projects. Later on, Davis et al. [3] refuted, revisited and revised the analysis developed by Tindell et al. In [4], Davis et al. extended the analysis in [2], [3] which is applicable to the CAN network where some nodes implement priority queues and some implement FIFO queues. The analysis in [2], [3] assumes that the CAN controllers have very large transmission buffers.

Some CAN controllers may have limited transmission buffers. If all such buffers in a CAN controller are occupied by lower priority messages, a higher priority message released in the same controller may suffer from priority inversion [2], [5], [6]. If the controller supports transmission abort requests then the lowest priority message in the transmission buffer (that is not under transmission) is swapped with the higher priority message from the message queue. During the swapping process, a lower priority message from the transmission buffer in any other controller may win the bus arbitration and contribute an extra delay in the blocking of the higher priority message. The copying delay and the extra blocking delay during the swapping process should be taken into account while calculating the response time of the higher priority message. Khan et al. [5] integrated this extra delay with the analysis in [2], [3] caused by priority inversion due to transmission abort requests supported in CAN controllers. According to [5], most of the CAN enabled ECUs are capable of aborting transmission requests.

However, the analysis in [2], [3], [4], [5] does not support response-times computation of mixed messages in CAN, i.e., the messages that are simultaneously time and event triggered. In [7], Mubeen et al. extended the existing analysis [2], [3] for mixed messages in CAN. The extended analysis supports the Worst Case Response Time (WCRT) computation of CAN messages that are queued for transmission periodically, sporadically and both periodically and sporadically (mixed). This analysis has been implemented in the existing industrial tool suite, i.e., Rubus-ICE [8], [9], [10]. In [11], [12], Mubeen et al. further extended the previous analysis [7] by integrating it with the analysis in [4] to support response time computation of mixed messages in CAN with priority- and FIFO-queued nodes. However, the analysis in [7], [11], [12] does not consider the extra timing overhead in the message response times caused by the priority inversion due to transmission abort requests in the CAN controllers.

B. Paper Contribution

We extend the existing analysis for mixed messages in CAN [7] by integrating the extra timing overhead in message response times caused by the priority inversion due to transmission abort requests supported in the CAN controllers. This overhead was first introduced in [2] and later derived in [5]. Mixed messages represent a common message transmission pattern which is implemented by some high-level protocols used in the industry. The extended analysis is generally applicable to any high-level protocol for CAN that uses periodic, sporadic, and mixed transmission of messages and supports transmission abort requests in CAN controllers.

II. Mixed Transmission Patterns Supported 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.

A. Implementation Method 1: CANopen Protocol

The CANopen protocol [13] supports mixed transmission mode that corresponds to the Asynchronous Transmission Mode coupled with the Event Timer. The Event Timer is used to transmit an asynchronous message cyclically. A mixed message can be queued for transmission at the arrival of an event provided the Inhibit Time has expired. The Inhibit Time is the minimum time that must be allowed to elapse between

the queueing of two consecutive messages. A mixed message can also be queued periodically at the expiry of the Event Timer. The Event Timer is reset every time the message is queued. Once a mixed message is queued, any additional queueing of the same message will not take place during the Inhibit Time [13]. The transmission pattern of a mixed message in CANopen is illustrated in Fig. 1. Message 1 is queued as soon as an event A arrives. Both the Event Timer and Inhibit Time are reset. As soon as the Event Timer expires, message 2 is queued due to periodicity and both the Event Timer and Inhibit Time are reset again. Similarly, message 3 is queued due to the expiry of the Event Timer. When an event B arrives, message 4 is immediately queued because the Inhibit Time has already expired. Note that the Event Timer is also reset at the same time when message 4 is queued as shown in Fig. 1. Message 5 is queued because of the expiry of Event Timer. Hence, there exists a dependency relationship between the Inhibit Time and the Event Timer.

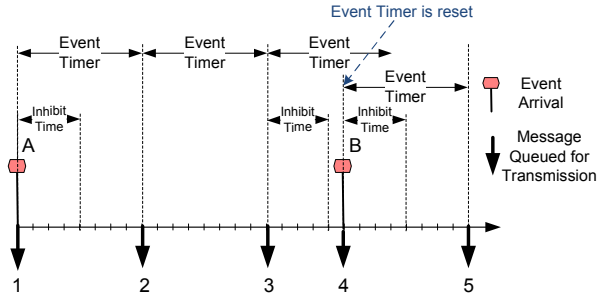

Fig. 1. Mixed transmission pattern in CANopen

B. Implementation Method 2: AUTOSAR Communication

AUTOSAR [14] can be viewed as a high-level protocol if it uses CAN for network communication. Mixed transmission mode in AUTOSAR is widely used in practice. A mixed message can be queued for transmission repeatedly with a period equal to the mixed transmission mode time period. The mixed message can also be queued at the arrival of an event provided the Minimum Delay Time (MDT) has been expired. However, each transmission of a mixed message, regardless of being periodic or sporadic, is limited by MDT. This means that both periodic and sporadic transmissions are delayed until MDT expires. The transmission pattern of a mixed message implemented by AUTOSAR is illustrated in Fig. 2.

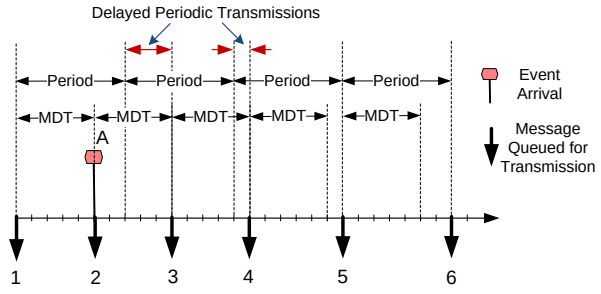

Fig. 2. Mixed transmission pattern in AUTOSAR

Message 1 is queued (MDT is started) because of partly periodic nature of a mixed message. When an event A arrives, message 2 is queued immediately because MDT has already expired. The next periodic transmission is scheduled 2 time units after the transmission of message 2. However, next two periodic transmissions corresponding to messages 3 and 4 are delayed because MDT is not expired. This is indicated by "Delayed Periodic Transmissions" in Fig. 2. The periodic transmissions corresponding to messages 5 and 6 take place at the scheduled time because MDT is already expired.

C. Implementation Method 3: HCAN Protocol

A mixed message defined by HCAN protocol [15] contains periodic and sporadic signals. A mixed message 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 is depicted in Fig. 3.

Message 1 is queued because of periodicity. As soon as event A arrives, message 2 is queued. 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 an 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 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.

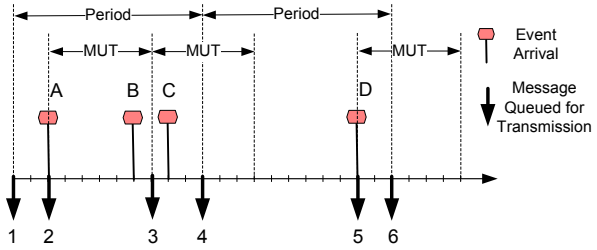

Fig. 3. Mixed transmission pattern in HCAN

D. Discussion

In the first method, the Event Timer is reset every time a mixed message is queued for transmission. The implementation method 2 is similar to method 1 to some extent. The main difference is that in method 2, the periodic transmission can be delayed until the expiry of MDT. Whereas in method 1, the periodic transmission is not delayed, in fact, the Event Timer is restarted with every sporadic transmission. The MDT timer is started with every periodic or sporadic transmission of a mixed message. Hence, the worst-case periodicity of a mixed message in methods 1 and 2 can never be higher than Inhibit Timer and MDT respectively. Therefore, the existing analyses [2], [3], [5] hold good. However, the periodic transmission is independent of the sporadic transmission in the third method. A mixed message can be queued for transmission even if MUT is not expired. Hence, the worst-case periodicity of a mixed message is neither bounded by period nor by MUT. Therefore, the analysis in [2], [3], [5] cannot be used for analyzing mixed messages in the third method.

III. System Scheduling Model

The system scheduling model is an extension of the model developed by Tindell et al. [2]. It combines the system

model of RTA of CAN for mixed messages [7] with the scheduling model in [5]. 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. Let K_c denote the transmission buffer 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 $\xi(m)$ can be 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 frame has a period, denoted by T_m. Whereas in the case of sporadic transmission, each frame has a MUT_m that refers to the minimum time that should elapse between the transmission of any two sporadic frames. Each message has a blocking time (B_m) which refers to the largest amount of time this message can be blocked by any lower priority message. Each message has a worst-case response time, denoted by R_m, and 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 (on the destination node).

Let CT_m denotes the maximum time required to swap a higher priority message m with the lowest priority message in the transmission buffer. The additional delay due to priority inversion (as discussed in Section I-A) is denoted by AD_m. We duplicate a message when its transmission type is mixed and treat it as two separate messages, i.e., periodic and sporadic. All the 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 deadline D_m. We assume that the deadlines can be greater than the periods or MUTs. We further assume that CAN controllers are capable of buffering more than one instance of a message.

IV. 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 [5]. Due to lack of space, we only discuss the extended analysis in the case when m is mixed and subjected to priority inversion.

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 by the following equations.

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

Using the existing analysis of mixed messages [7], WCRTs of each instance of m_P and m_E are given by:

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

$$R_{m_E}(q_{m_E}) = J_m + \omega_{m_E}(q_{m_E}) - q_{m_E}MUT_m + C_m \tag{5}$$

C_m in (4) and (5) 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 and m_E is denoted by ω_{m_P} and ω_{m_E} in (4) and (5) respectively. ω_{m_P} and ω_{m_E} consist of three factors.

1) The blocking time and the time to swap m_P or m_E with the lowest priority message in K_c, i.e., B_m + CT_m.
2) Interference from higher priority messages in the system.
3) Self interference, i.e., m_P can be interfered by m_E and vice versa.

ω_{m_P} and ω_{m_E} can be computed by integrating the existing analysis for mixed messages [7] with [5] as shown below.

$$\omega_{m_P}^{n+1}(q_{m_P}) = \hat{\mathbf{B}}_m + CT_m + q_{m_P}C_m + \sum_{\forall k \in hp(m)} I_{k_P}C_k + Q_{m_E}^P C_m \tag{6}$$

$$\omega_{m_E}^{n+1}(q_{m_E}) = \hat{\mathbf{B}}_m + CT_m + q_{m_E}C_m + \sum_{\forall k \in hp(m)} I_{k_E}C_k + Q_{m_P}^E C_m \tag{7}$$

C_m can be selected as the initial value of the queueing delay. I_{k_P} and I_{k_E} are given by (8) and (10) respectively.

$$I_{k_P} = \begin{cases} \left\lceil \frac{\omega_{m_P}^n(q_{m_P}) + \hat{\mathbf{J}}_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(k) = P \\ \left\lceil \frac{\omega_{m_P}^n(q_{m_P}) + \hat{\mathbf{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = S \\ \left\lceil \frac{\omega_{m_P}^n(q_{m_P}) + \hat{\mathbf{J}}_k + \tau_{bit}}{T_k} \right\rceil + \left\lceil \frac{\omega_{m_P}^n(q_{m_P}) + \hat{\mathbf{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = M \end{cases} \tag{8}$$

It is evident from (8) and (10) that m_P and m_E receive double interference from every higher priority mixed message. Note that the jitter J_k is replaced with increased jitter $\hat{\mathbf{J}}_k$ in (8) and (10) compared to the existing analysis [7]. This is because the additional delay received by a 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.

$$\hat{\mathrm{J}}_k = J_k + AD_k \tag{9}$$

$$I_{k_E} = \begin{cases} \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + \hat{\mathrm{J}}_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{P} \\ \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + \hat{\mathrm{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{S} \\ \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + \hat{\mathrm{J}}_k + \tau_{bit}}{T_k} \right\rceil + \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + \hat{\mathrm{J}}_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if} \xi(\mathrm{k})=\mathrm{M} \end{cases} \tag{10}$$

$\hat{\mathrm{B}}_m$ in equations (6) and (7) can be computed, according to [5], by adding additional delay of m to its blocking time.

$$\hat{\mathrm{B}}_m = B_m + AD_m;\ where,\ B_m = \max_{\forall k \in lp(m)}(C_k) \tag{11}$$

The additional delay for m is adapted from [5] as follows.

$$AD_m = max(0, \max_{\forall l \in \aleph_c \wedge P_l > P_m}(CT_l) + \max_{P_m < P_l \leq P_j}(C_l) - B_m) \tag{12}$$

Where the priority of message j is lower than m and higher than the highest priority message in K_c.

Effect of self interference. The effect of self interference can be seen in the last terms of (6) and (7). $Q^P_{m_E}$ denotes the total number of instances of m_E that are queued ahead of $q^{th}_{m_P}$ instance of m_P. Similarly, $Q^E_{m_P}$ denotes the total number of instances of m_P that are queued ahead of $q^{th}_{m_E}$ instance of m_E. We will reuse the effect of self interference that we derived in [7] with a slight modification (i.e., J_m will be replaced with $\hat{\mathrm{J}}_m$). The values of $Q^P_{m_E}$ and $Q^E_{m_P}$ are computed as follows.

$$Q^P_{m_E} = \left\lceil \frac{q_{m_P} T_m + \hat{\mathrm{J}}_m}{MUT_m} \right\rceil; Q^E_{m_P} = \left\lceil \frac{q_{m_E} MUT_m + \hat{\mathrm{J}}_m}{T_m} \right\rceil \tag{13}$$

Length of the busy period. The length of priority level-m busy period, denoted by t_m, can be computed by adapting the existing analysis [7].

$$t^{n+1}_m = \hat{\mathrm{B}}_m + \sum_{\forall k \in hep(m)} I'_k C_k \tag{14}$$

I'_k in (14) 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.

$$I'_k = \begin{cases} \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{T_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{P} \\ \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{S} \\ \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{T_k} \right\rceil + \left\lceil \frac{t^n_m + \hat{\mathrm{J}}_k}{MUT_k} \right\rceil, & \text{if } \xi(\mathrm{k}) = \mathrm{M} \end{cases} \tag{15}$$

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 (14). Therefore, there is no need to compute t_m for m_P and m_E separately. t_m should be computed only once for a mixed message m. In order to solve the recursive equation (14), C_m can be used as an initial value of t^n_m. Although the length of the busy period is the same for m_P and m_E, the number of instances of both the messages that become ready for transmission just before the end of busy period, i.e., Q_{m_P} and Q_{m_E} respectively, may be different. This is because the computation of Q_{m_P} and Q_{m_E} require T_m and MUT_m respectively and which may have different values. Q_{m_P} and Q_{m_E} can be computed as follows.

$$Q_{m_P} = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil;\ Q_{m_E} = \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil \tag{16}$$

V. Summary

The existing response-time analysis for mixed messages in CAN assumes that CAN controllers have large transmission buffers. However, some CAN controllers may have limited-space transmission buffers. Due to this hardware limitation, higher priority messages may undergo priority inversion which can contribute an extra delay to their response times. Mixed messages are implemented by several high-level protocols for CAN used in the industry today. We extended the existing analysis to support mixed messages in CAN network where CAN controllers support transmission abort requests in the transmission buffers. We plan to implement the extended analysis in the existing industrial tool suite (Rubus-ICE) and conduct an industrial case study.

Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) within the project FEMMVA. We would like to thank the industrial partners Arcticus Systems, BAE Systems Hägglunds and Volvo Construction Equipment, Sweden.

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. 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] "Arcticus Systems," http://www.arcticus-systems.com.

[10] 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.

[11] 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.

[12] 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.

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

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

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