

# Response Time Analysis with Offsets for Mixed Messages in CAN with Buffer Limitations\*

Saad Mubeen\*<sup>†</sup>, Jukka Mäki-Turja\*<sup>†</sup> and Mikael Sjödin\*

\*Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Sweden

<sup>†</sup>Arcticus Systems, Järfälla, Sweden

{saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se

**Abstract.** The existing worst-case response-time analysis for Controller Area Network (CAN) does not support mixed messages that are scheduled with offsets in the systems where the CAN controllers implement abortable transmit buffers. Mixed messages are partly periodic and partly sporadic. These messages are implemented by several higher-level protocols based on CAN that are used in the automotive industry. Moreover, most of the CAN controllers implement abortable transmit buffers. We extend the existing analysis with offsets for mixed messages in CAN. The extended analysis is applicable to any higher-level protocol for CAN that uses periodic, sporadic, and mixed transmission of messages where periodic and mixed messages can be scheduled with offsets in the systems that implement abortable transmit buffers in the CAN controllers.

## 1 Introduction

Controller Area Network (CAN) [1] is a widely used real-time network protocol in the automotive domain. According to an estimate by CAN in Automation [2], more than two billion CAN enable controllers have been sold mostly in the automotive domain. In 2003, CAN was standardized as ISO 11898-1 [3]. It is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 Mbit/s. There are several higher-level protocols for CAN that are developed for many industrial applications such as CAN Application Layer (CAL), CANopen, J1939, Hägglunds Controller Area Network (HCAN), CAN for Military Land Systems domain (MilCAN). Often, CAN is employed in predictable and safety-critical systems. The providers of these systems are required to ensure that the systems meet their deadlines. For this purpose, several *a priori* timing analysis techniques including Response-Time Analysis (RTA) [4–7] have been developed by the research community. RTA is a powerful and well established method to calculate upper bounds on the response times of tasks or messages in a real-time system or a network respectively.

### 1.1 Motivation and related work

Tindell et al. [8] developed RTA for CAN with priority queues. It has been implemented in the analysis tools that are used in the automotive industry, e.g., VNA [9].

---

\* **This report is an unpublished work. A version of it is under review. The sole purpose of this report is to provide reference for another work under review**

Davis et al. [10] refuted, revisited and revised the seminal analysis of [8]. The revised analysis is implemented in the existing industrial tool suite Rubus-ICE [11, 12]. However, these analyses do not support the network where CAN controllers<sup>1</sup> implement abortable transmit buffers, e.g., Atmel AT89C51CC03/AT90CAN32/64 and Microchip MPC2515 [13]. In order to correctly calculate the response times of CAN messages, these type of practical limitations in the CAN controllers should be considered in RTA [14, 15]. Khan et al. [16] extended the revised seminal analysis for the network where nodes implement abortable transmit buffers. In [17, 18, 13], the previous RTA [8, 10] is extended to support CAN network where nodes implement priority, FIFO and work-conserving queues. But, none of the analysis discussed above supports messages that are scheduled with offsets, i.e., using externally imposed delays between the times when the messages can be queued. The worst-case RTA for CAN messages with offsets has been developed in several works [19–23].

All of the above analyses assume that messages are queued for transmission periodically or sporadically. They do not support mixed messages in CAN which are partly periodic and partly sporadic. Mixed messages are implemented by several higher-level protocols based on CAN that are used in the automotive industry. Mubeen et al. [24] extended the existing analysis to support mixed messages in CAN where nodes implement priority queues. Mubeen et al. [25] further extended their analysis to support mixed messages in CAN with FIFO queues. In [26], Mubeen et al. presented work in progress for the extension of RTA for mixed messages in CAN with abortable transmit buffers. In [27], we extended the existing analysis for CAN [20] to support mixed messages that are scheduled with offsets. However, this analysis is restricted due to limitations regarding message jitter and deadlines. In [28], we removed these limitations and extended the analysis for mixed messages [24] by building it upon the analysis for CAN messages with offsets [23]. In this paper, we extend our work-in-progress paper [26] to support RTA with offsets for mixed messages in CAN where the CAN controllers implement abortable transmit buffers. Fig. 1 depicts the relation between the existing and extended analyses.

## 1.2 Previous work and paper contribution

We extend worst-case response-time analysis of CAN to support the analysis of mixed messages that are scheduled with offsets in the system where CAN controllers implement abortable transmit buffers. The existing analysis for mixed messages with offsets [27, 28] does not support transmission abort requests in the CAN controllers. The extended analysis is build upon our previous work (work-in-progress paper) [26]. Since the release jitter can be higher than message period, e.g., in gateway nodes, the extended analysis assumes arbitrary jitter and deadline. This means each one of them can be lower, equal or higher than the transmission period of the message. The extended analysis is applicable to any higher-level protocol for CAN that uses periodic, sporadic and mixed transmission of messages; whereas the periodic and mixed message can be scheduled with offsets.

<sup>1</sup> For convenience, we overload the terms node and CAN controller

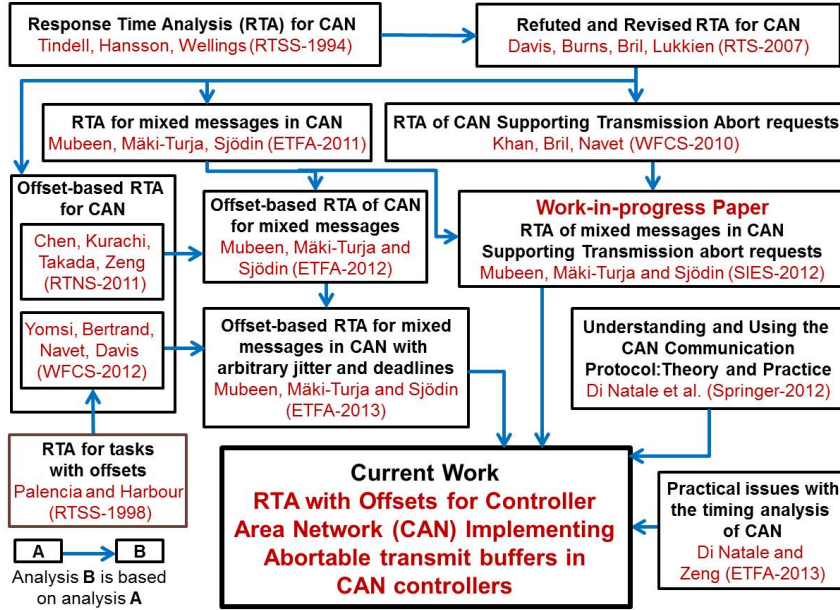


Fig. 1. Relation between the existing and extended Response Time Analyses

## 2 Mixed messages implemented by the higher-level protocols

In this section, we discuss and compare the implementation of mixed messages by several higher-level protocols for CAN that are used in the automotive industry. Traditionally, it is assumed that the tasks queueing CAN messages are invoked either periodically or sporadically. If a message is periodically queued for transmission, we use the term “Period” to refer to its periodicity. A sporadic message is queued for transmission as soon as an event occurs that changes the value of one or more signals contained in the message provided the Minimum Update Time ( $MUT^2$ ) between the queueing of two successive sporadic messages has elapsed. However, there are some higher-level protocols for CAN used in the industry that support queueing of messages periodically as well as sporadically. These messages are said to be mixed, i.e., a mixed message is simultaneously time- and event-triggered.

### 2.1 Method 1: Implementation in the CANopen protocol

The CANopen protocol [29] supports mixed transmission that corresponds to the Asynchronous Transmission Mode coupled with the Event Timer. 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

<sup>2</sup> We overload the term “ $MUT$ ” to refer to the Inhibit Time in the CANopen protocol [29] and the Minimum Delay Time (MDT) in the AUTOSAR communication [30].

the queuing of two consecutive messages. A mixed message can also be queued periodically when the Event Timer expires. The Event Timer is reset every time the message is queued. Once a mixed message is queued, any additional queuing of this message will not take place during the Inhibit Time [29]. The transmission pattern of a mixed message in CANopen is illustrated in Fig. 2(a). The down-pointing arrows symbolize the queuing of messages while the upward lines (labeled with alphabetic characters) represent arrival of the events. Message 1 is queued as soon as the 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. When the event *B* arrives, message 3 is immediately queued because the Inhibit Time has already expired. Note that the Event Timer is also reset at the same time when message 3 is queued as shown in Fig. 2(a). Message 4 is queued because of the expiry of the Event Timer. There exists a dependency relationship between the Inhibit Time and the Event Timer, i.e., the Event Timer is reset with every sporadic transmission.

## 2.2 Method 2: Implementation in AUTOSAR

AUTOSAR [31] can be viewed as a higher-level protocol if it uses CAN for network communication. Mixed transmission mode in AUTOSAR is widely used in practice. In AUTOSAR, 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 the mixed message, regardless of being periodic or sporadic, is limited by the *MDT*. This means that both periodic and sporadic transmissions are delayed until the *MDT* expires. The transmission pattern of a mixed message implemented by AUTOSAR is illustrated in Fig. 2(b). Message 1 is queued (the *MDT* is started) because of partly periodic nature of the mixed message. When the event *A* arrives, message 2 is queued immediately because the *MDT* has already expired. The next periodic transmission is scheduled 2 time units after the transmission of message 2. However, the next two periodic transmissions corresponding to messages 3 and 4 are delayed because the *MDT* is not expired. This is indicated by the comment “Delayed Periodic Transmissions” in Fig. 2(b). The periodic transmissions corresponding to messages 5 and 6 occur at the scheduled times because the *MDT* is already expired in both cases.

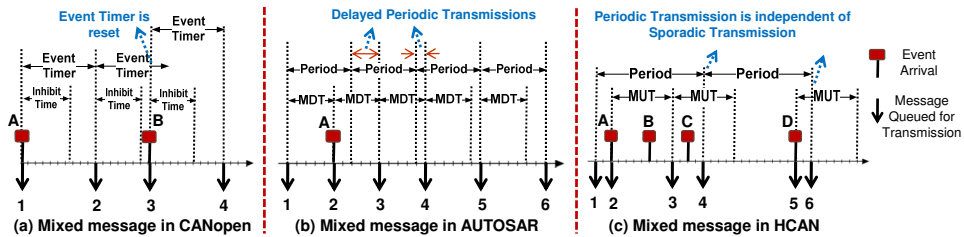


Fig. 2. Mixed transmission pattern in higher-level protocols for CAN

### 2.3 Method 3: Implementation in the HCAN protocol

A mixed message in HCAN protocol [32] contains signals out of which some are periodic and some are sporadic. 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 the  $MUT$  between the queuing of two successive sporadic instances of the mixed message has elapsed. Hence, the transmission of the mixed message due to arrival of events is constrained by the  $MUT$ . The transmission pattern of the mixed message is illustrated in Fig. 2(c). 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 the  $MUT$  is not expired yet. As soon as the  $MUT$  expires, message 3 is queued. Message 3 contains the signal changes that correspond to event  $B$ . Similarly, a message is not immediately queued when the event  $C$  arrives because the  $MUT$  is not expired. Message 4 is queued because of the periodicity. Although, the  $MUT$  was not 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 the  $MUT$  expires. This indicates that the periodic transmission of the mixed message cannot be interfered by its sporadic transmission (a unique property of the HCAN protocol). When the event  $D$  arrives, a sporadic instance of the mixed message is immediately queued as message 5 because the  $MUT$  has already expired. Message 6 is queued due to periodicity.

### 2.4 Discussion

In the first method, the Event Timer is reset every time the mixed message is queued for transmission. The implementation of mixed message in 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 the  $MDT$ . 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 the mixed message. Hence, the worst-case periodicity of the mixed message in methods 1 and 2 can never be higher than the Inhibit Timer and  $MDT$  respectively. This means that the models of mixed messages in the first and second implementation methods reduce to the classical sporadic model. Therefore, the existing analyses for CAN messages with offsets [20, 21, 19, 22, 23, 16] can be used for analyzing mixed messages in the first and second implementation methods.

However, periodic transmission is independent of the sporadic transmission in the third method because the periodic timer is not reset with every sporadic transmission. The message can be queued for transmission even if the  $MUT$  is not expired. Hence, the worst-case periodicity is neither bounded by period nor by the  $MUT$ . Therefore, the analyses in [20, 21, 19, 22, 23, 16] cannot be used for analyzing the mixed messages in the third implementation method.

## 3 Effect of abortable transmit buffers on RTA

When there are fewer number of transmit buffers in a CAN controller compared to the number of messages sent by the ECU, the messages may be subjected to extra delay

and jitter due to priority inversion. Most of the CAN controllers support transmission abort requests, e.g., Atmel AT89C51CC03/AT90CAN32/64 and Microchip MPC2515 [13, 16]. If a CAN controller supports transmission abort requests (and implements at least 3 transmit buffers) then the lowest priority message in the transmit 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 transmit buffer in any other controller may win the bus arbitration and start its transmission. This causes priority inversion for the higher priority message. As a result, it contributes an extra delay to the response time of the higher priority message. The copying delay and the extra blocking delay should be taken into account while calculating the response time of the higher priority message; otherwise, the calculated response times can be optimistic.

### 3.1 Additional delay and jitter due to priority inversion

In order to demonstrate the additional delay due to priority inversion when CAN controllers implement abortable transmit buffers, consider an example in Fig. 3. Assume there are three nodes  $CC_c$ ,  $CC_j$  and  $CC_k$  in the system and each node has three transmit buffers.  $m_1$  is the highest priority message in  $CC_c$  as well as in the system. When  $m_1$  becomes ready for transmission in the message queue of  $CC_c$ , a lower priority message  $m_6$  belonging to  $CC_k$  is already under transmission.  $m_6$  cannot be preempted because CAN uses fixed priority non-preemptive scheduling. This represents the blocking delay for  $m_1$ . At this point in time, all transmit buffers in  $CC_c$  are occupied by the lower priority messages (say  $m_3$ ,  $m_4$  and  $m_5$ ). The device drivers signal an abort request for the lowest priority message in  $K_c$  (transmit buffers in  $CC_c$ ) that is not under transmission.

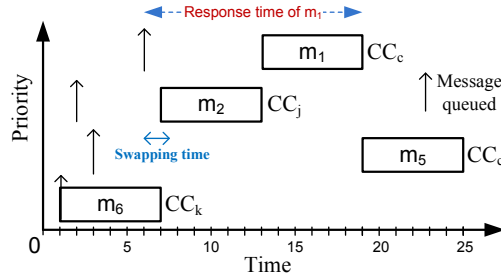


Fig. 3. Priority inversion in the case of abortable transmit buffers

Hence,  $m_5$  is aborted and copied from the transmit buffer to the message queue, whereas  $m_1$  is moved to the vacated transmit buffer. The time required to do this swapping is identified as *swapping time* in Fig. 3. During the swapping time, a series of events may occur:  $m_6$  finishes its transmission, new arbitration round starts, another message  $m_2$  belonging to node  $CC_j$  and having priority lower than  $m_1$  wins the arbitration and starts its transmission. Thus  $m_1$  has to wait in the transmit buffer until  $m_2$  finishes its transmission. This results in the priority inversion for  $m_1$  and adds an extra

delay to its response time. In [16], Khan et al. pointed out the extra delay of the higher priority message appears as additional jitter to the lower priority messages, e.g.,  $m_5$  in Fig. 3.

### 3.2 Discussion on message copy time and delay

If the message copy time is smaller than or equal to the inter-frame space (i.e., time to transmit 3 bits on CAN bus or  $3*\tau_{bit}$  time), a lower priority message in the transmit buffer (that is not under transmission) can be swapped with a higher priority message in the message queue before transmission of the next frame [1]. Hence, there will be no priority inversion. This means that the message copy time must be, at least,  $4*\tau_{bit}$  for the priority inversion to occur. In Legacy systems, there may be slow controllers, i.e., the speed of the controllers can be slower than the maximum operating speed of the CAN bus (1 Mbit/s). Since the amount of data transmitted in a CAN message ranges from 0 to 8 bytes, the transmission time of a message also varies accordingly. According to [10], the transmission time of a CAN message with standard frame format ranges from  $55*\tau_{bit}$  to  $135*\tau_{bit}$  for the amount of data contained in the message that ranges from 0 to 8 bytes respectively. Intuitively, the message copy time of  $4*\tau_{bit}$  can range from 7.3% to 3% of transmission time of a message with 0 to 8 bytes of data respectively. Due to slow controllers in legacy systems, the message copy time can be greater than  $4*\tau_{bit}$ , hence, higher than 7.3% of its transmission time.

### 3.3 Messages safe from priority inversion

It should be noted that not all messages in a node suffer from priority inversion [16]. The number of lowest priority messages equal to the number of transmit buffers in a node will be safe from priority inversion. Whereas the rest of the messages in the same node may suffer from priority inversion. E.g., let there be 4 transmit buffers in a node that sends six messages  $m_1, m_2, m_3, m_4, m_5$  and  $m_6$ .  $m_1$  has the highest priority while  $m_6$  has the lowest priority. Assume  $m_4$  arrives in the message queue when 3 out of 4 transmit buffers are occupied by the three lowest priority messages  $m_6, m_5$  and  $m_4$ . The fourth transmit buffer can either be empty or occupied by one of the higher priority messages  $m_1$  or  $m_2$ . If the fourth transmit buffer is empty then  $m_4$  is immediately copied to it. On the other hand,  $m_4$  has to wait in the message queue because at least one transmit buffer contains a higher priority message. In both cases there is no need to abort any transmission. This implies that  $m_6, m_5, m_4$  and  $m_3$  will be safe from priority inversion, whereas  $m_1$  and  $m_2$  may face it.

## 4 System model

The system consists of a number of CAN controllers, denoted by  $CC_1, CC_2, \dots, CC_n$ , that are connected to a single CAN network. The nodes implement priority queues which means that the highest priority message in a node enters into the bus arbitration. We assume that each CAN controller has a finite number of transmit buffers (however,

not less than three). Let the transmit buffers in a CAN controller  $CC_c$  be represented by  $K_c$ . The number of transmit buffers in  $CC_c$  can be found using the function  $Sizeof(K_c)$ .

Each CAN message  $m_m$  has a unique identifier and a priority denoted by  $ID_m$  and  $P_m$  respectively. The priority of a message is assumed to be equal to its ID. The priority of  $m_m$  is considered higher than the priority of another message  $m_n$  if  $P_m < P_n$ . Let the sets  $hp(m_m)$ ,  $lp(m_m)$ , and  $hep(m_m)$  contain the messages with priorities higher, lower, and equal and higher than  $m_m$  respectively. No doubt, priorities of CAN messages are unique, the set  $hep(m_m)$  is used in the case of mixed messages. The *FRAME\_TYPE* attribute specifies whether a frame is a standard or an extended CAN frame. The standard CAN frame uses an 11-bit identifier whereas the extended CAN frame uses a 29-bit identifier. We define a function  $\xi_m$  that denotes transmission type of a message.  $\xi_m$  specifies whether  $m_m$  is periodic ( $P$ ), sporadic ( $S$ ) or mixed ( $M$ ). Formally, the domain of  $\xi_m$  can be defined as follows.

$$\xi_m \in [P, S, M]$$

The transmission time of  $m_m$  is denoted by  $C_m$ . Each message can carry a data payload denoted by  $s_m$  that ranges from 0 to 8 bytes. In the case of periodic transmission,  $m_m$  has a period which is denoted by  $T_m$ . Whereas in the case of sporadic transmission,  $m_m$  has the  $MUT_m$  (Minimum Update Time) that refers to the minimum time that should elapse between the transmission of any two sporadic messages. The queueing jitter of  $m_m$  is denoted by  $J_m$  which is inherited from the task that queues it. We assume that  $J_m$  can be smaller, equal or greater than  $T_m$  or  $MUT_m$ .  $B_m$  denotes the blocking time of  $m_m$  which refers to the largest amount of time  $m_m$  can be blocked by any lower priority message.

We duplicate a message when its transmission type is mixed. Each mixed message  $m_m$  is duplicated; which means it is treated as two separate messages, i.e., one periodic and the other sporadic. These duplicates share all attributes except for  $T_m$  and  $MUT_m$ . The periodic copy inherits  $T_m$ ; whereas, the sporadic copy inherits the  $MUT_m$ . The worst-case response time of  $m_m$  is denoted by  $R_m$ . It is defined as the longest time between the queueing of  $m_m$  in the sending node and its delivery to the destination buffer in the destination node.  $m_m$  is considered schedulable if its  $R_m$  is less than or equal to its deadline  $D_m$ . The system is considered schedulable if all messages are schedulable. We consider arbitrary deadlines which means they can be greater than the periods or  $MUT$ s of corresponding messages. We assume that the CAN controllers are capable of buffering more than one instance of a message. All instances of a message are considered to be transmitted in the same order in which they are queued (we assume FIFO policy among the instances of the same message).

Let the offset of  $m_m$  be denoted by  $O_m$ . We assume that the offset of a message is always smaller than its period. The first arrival time of  $m_m$  is equal to its offset; whereas, the subsequent arrivals occur periodically with respect to the first. The smallest offset in a node is assumed to be equal to zero. It is important to note that each node has its own local time and there is no global synchronization among the nodes. We assume that the offset relations exist only among periodic messages and periodic copies of mixed messages within a node. Hence, there are no offset relations: (1) among sporadic messages, (2) between a periodic message and a sporadic message, (3) between a



periodic copy of a mixed message and a sporadic message, (4) between duplicates of a mixed message, (5) between any two messages from different nodes.

All periodic messages and periodic copies of mixed messages in a node are collected together in a single transaction denoted by  $T_i$ . Each transaction belongs to  $\Gamma$  which is the set of all transactions in the system. This transactional model is adapted from [33]. It should be noted that the offset relations exist only within a transaction, and there are no offset relations among any two transactions. Within context of a transaction, we denote a message  $m_j$  belonging to transaction  $T_i$  by  $m_i^j$ . The period of  $T_i$  is denoted by  $T_{T_i}$  and is defined as the Least Common Multiple (LCM) of the periods of all messages belonging to  $T_i$ . Each sporadic message or sporadic copy of a mixed message is modeled as a separate transaction.

Consider a simple example shown in Fig. 4. A node transmits two messages: a mixed message  $m_1$  with high priority and a periodic message  $m_2$  with low priority. Transaction  $T_1$  contains both  $m_2$  and periodic copy of  $m_1$ . The period of  $T_1$  denoted by  $T_{T_1}$  is the LCM of  $T_1$  and  $T_2$ . Transaction  $T_2$  consists of only sporadic copy of  $m_1$ .

## 5 Extended worst-case response-time analysis

Let the message under analysis be denoted by  $m_m$  and it belongs to node  $CC_i$ . Since  $m_m$  may or may not suffer from priority inversion, we consider two different cases for calculating its response time by adapting the analysis in [16]. However, in each case,  $m_m$  is analyzed differently based on its transmission type. Intuitively, in each case, we consider three different sub-cases namely periodic, sporadic and mixed. Let us first discuss few terms that are used in the analysis.

**Maximum Busy Period.** In order to calculate the worst-case response time of  $m_m$ , the maximum busy period [8, 10] for priority level- $m$  should be known first. It is the longest contiguous interval of time during which  $m_m$  is unable to complete its transmission due to two reasons. First, the bus is occupied by the higher priority messages. Second, a lower priority message already started its transmission when  $m_m$  is queued for transmission. The maximum busy period starts at the so-called critical instant.

**Critical Instant.** We redefine the critical instant for priority level- $m$  busy period as the instant when (1)  $m_m$  or any other higher priority message belonging to the same node as that of  $m_m$  is queued for transmission, (2) at least one message with priority higher than  $m_m$  is queued for transmission from every node, (3) all sporadic messages and sporadic copies of mixed messages belonging to the set  $hp(m_m)$  from every node are simultaneously queued for transmission at the respective nodes, and (4) a lower priority message just started its transmission when  $m_m$  is queued. The critical instant for priority level-2 busy period is identified at  $t_c$  in Fig. 3. According to condition (3), the arrival of  $T_2$  should coincide with the critical instant.

**Worst-Case Candidates.** The main issue regarding condition (2) is to determine which message in the set  $hp(m_m)$  is the candidate to start the critical instant, i.e., contributing to the worst-case response-time of  $m_m$ . The solution is that any message in the set  $hp(m_m)$  can be the worst-case candidate. Therefore, each message has to be tested in the busy period as the potential worst-case candidate. The response time of  $m_m$  should be calculated from every worst-case candidate and the maximum among all should be

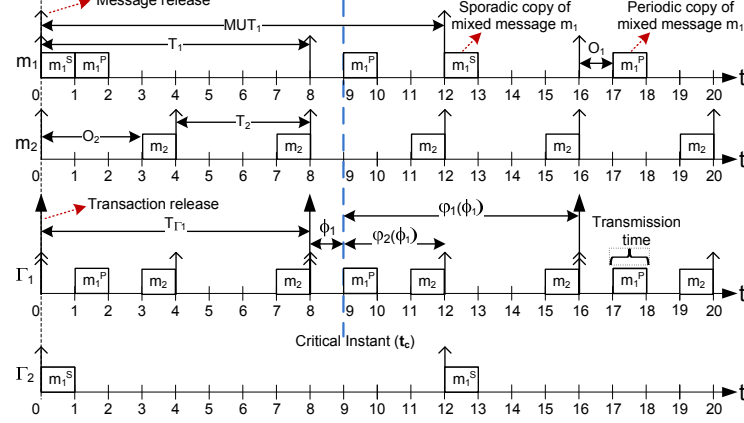


Fig. 4. Example of the transactional model

considered as the worst-case response time of  $m_m$ . In this work, we present RTA with respect to any worst-case candidate.

**Calculations for the additional jitter.** These calculations are adapted from the analysis in [16]. Let  $K_i$  denote the transmit buffer queue in  $CC_i$ . Let  $CT_m$  denotes the maximum between the time required to copy  $m_m$  from the message queue to the transmit buffer and from transmit buffer to the message queue. As noted in [16], these two times are very similar to each other. Let the additional jitter of  $m_m$  as seen by the lower priority messages due to priority inversion be denoted by  $AJ_m$ . Where  $AJ$  stands for “Additional Jitter”. The maximum jitter of  $m_m$  denoted by  $\hat{J}_m$  is the summation of its original jitter  $J_m$  and the additional jitter due to priority inversion. Mathematically, the additional jitter of  $m_m$  that is seen by lower priority messages is calculated as follows.

$$\hat{J}_m = J_m + AJ_m \quad (1)$$

The additional jitter for  $m_m$  depends upon three elements: (1) the largest copy time of a message in the set of lower priority messages that belong to the same node  $CC_i$ ; (2) the largest value among the worst-case transmission times of all those messages whose priorities are lower than the priority of  $m_m$  but higher than the highest priority message in  $K_i$ ; and (3) since the original blocking time  $B_m$  for  $m_m$  is separately considered as part of the queuing delay, it should be subtracted from the additional delay.

Therefore,  $AJ_m$  is calculated as follows:

$$AJ_m = \max(0, \max_{\forall m_l \in CC_i \wedge m_l \in lep(m_m)} (CT_l) + \max_{P_m < P_l \leq P_{h_{K_i}}} (C_l) - B_m) \quad (2)$$

where  $m_{h_{K_i}}$  is the highest priority message in  $K_i$ .

**Calculations for the blocking delay.** When  $m_m$  is subjected to priority inversion, it experiences an extra amount of blocking in addition to the original blocking delay  $B_m$ .

Let the total blocking delay for  $m_m$  due to priority inversion be denoted by  $\hat{B}_m$ . It is equal to the sum of the original blocking delay and the largest copy time of a message in the set of lower priority messages that belong to the same node  $CC_i$ .

$$\hat{B}_m = \max_{\forall m_j \in lep(m_m)} \{C_j\} + \max_{\forall m_l \in CC_i \wedge m_l \in lep(m_m)} (CT_l) \quad (3)$$

Since we consider arbitrary deadlines,  $m_m$  can also be blocked from its own previous instance due to push-through blocking [10]. That is the reason why (3) includes the function  $lep(m_m)$  instead of  $lp(m_m)$ .

### 5.1 Case 1: When message under analysis is subjected to priority inversion

**Case 1(a): When  $m_m$  is a periodic message** Let  $m_m$  belongs to transaction  $\Gamma_i$ . The worst-case response time of  $m_m$  is equal to the maximum value among the response times of all of its instances. We calculate the response times of all instances of  $m_m$  within priority level-m busy period. Let  $q_m$  denote the instances of  $m_m$ . Let  $q_m^L$  and  $q_m^H$  denote lowest- and highest-numbered instances respectively. The worst-case response time of  $m_m$  is given by:

$$R_m = \max\{R_m(q_m)\}, \quad \forall q_m^L \leq q_m \leq q_m^H \quad (4)$$

It should be noted that  $q_m$  is equal to 1 if the message instance is queued for transmission between the critical instant and  $T_m$ . Further,  $q_m$  is equal to 2 if the message instance is queued for transmission between  $T_m$  and  $2.T_m$ . Similarly,  $q_m$  is equal to 0 if the message instance is queued for transmission between the critical instant and  $-T_m$ . Since the jitter of a message can be greater than its transmission period, it is possible that the previous instances of the message may also be delayed due to jitter and enter in the maximum busy period. The calculations for the response time of instance  $q_m$  are adapted from [28, 23]. However, these calculations should consider three more elements: (1) copying delay  $CT_m$  for every instance of  $m_m$  in the priority level-m busy period; (2) additional jitter experienced by  $m_m$  due to higher priority messages; and (3) additional blocking delay as shown in (3).

$$R_m(q_m) = ST_m + C_m + CT_m - (\varphi_m(\phi_i) + (q_m - 1).T_m) \quad (5)$$

$\phi_i$  in (5) denotes the time interval between latest arrival of  $\Gamma_i$  (prior to the critical instant) and the critical instant. Consider the example message set in Fig. 3.  $\phi_i$  is equal to 1 time unit and is identified as  $\phi_1$  on the third time line from the top.  $\varphi_m(\phi_i)$  in (5) represents the length of the time interval between the critical instant and first release of  $m_m$  that occurs at or after the critical instant. Consider again the example message set in Fig. 3.  $\varphi_m(\phi_i)$  for messages  $m_1^P$  and  $m_2$  are identified by  $\varphi_1(\phi_1)$  and  $\varphi_2(\phi_1)$  respectively. The calculations for  $\varphi_m(\phi_i)$  are adapted from [33] as follows.

$$\varphi_m(\phi_i) = (T_m - (\phi_i - O_m) \bmod T_m) \bmod T_m \quad (6)$$

$ST_m$  in (5) denotes the Start Time (ST) when the priority level-m busy period ends and  $m_m(q_m)$  can start its transmission. Basically, it sums up the interferences due to

higher priority messages, previous instances of the same message and the blocking factor. It can be calculated by solving the following equation.

$$ST_m^{n+1} = \hat{B}_m + (q_m - q_m^L) \cdot C_m + (q_m - q_m^L) \cdot CT_m + \sum_{\forall \Gamma_k \in \Gamma} W_m(\Gamma_k, \phi_k, ST_m^n) \quad (7)$$

Where the terms  $(q_m - q_m^L) \cdot C_m$  and  $(q_m - q_m^L) \cdot CT_m$  represent the effect of interference and copy times of previous instances of  $m_m$  that are queued ahead of the instance under analysis. (7) is an iterative equation. It is solved iteratively until two consecutive solutions become equal. The starting value for  $ST_m^n$  in (7) can be selected equal to  $\hat{B}_m + (q_m - q_m^L) \cdot C_m + (q_m - q_m^L) \cdot CT_m$ . In (7),  $W_m$  represents the amount of interference due to the messages in the set  $hp(m_m)$  that are queued for transmission since the beginning of the busy period. It is important to mention that a message cannot be interfered by higher priority messages during its transmission because CAN uses fixed-priority non-preemptive scheduling. Whenever we use the term interference, it refers to the amount of time  $m_m$  has to wait in the send queue because the higher priority messages win the arbitration, i.e., the right to transmit before  $m_m$ .  $W_m$  can be calculated as follows.

$$W_m(\Gamma_k, \phi_k, ST_m^n) = \sum_{\forall m_j \in hp_k(m_m)} \Upsilon_k^j(ST_m^n) \cdot C_j \quad (8)$$

Where  $hp_k(m_m)$  represents the set of all those messages that belong to  $\Gamma_k$  and have priority higher than  $m_m$ .  $\Upsilon_k^j(ST_m^n)$  in (8) is calculated differently based on the transmission type  $\xi_j$  of the higher priority message  $m_j$ . The calculations for  $\Upsilon_k^j(ST_m^n)$  are adapted from [23] and [24] as follows.

$$\Upsilon_k^j(ST_m^n) = \begin{cases} \left\lfloor \frac{\hat{J}_j + \varphi_j(\phi_k)}{T_j} \right\rfloor + \left\lfloor \frac{ST_m^n - \varphi_j(\phi_k)}{T_j} \right\rfloor + 1, & \text{if } \xi_j = P \\ \left\lfloor \frac{ST_m^n + \hat{J}_j}{MUT_j} \right\rfloor + 1, & \text{if } \xi_j = S \\ \left\lfloor \frac{\hat{J}_j + \varphi_j(\phi_k)}{T_j} \right\rfloor + \left\lfloor \frac{ST_m^n - \varphi_j(\phi_k)}{T_j} \right\rfloor + 1 \\ + \left\lfloor \frac{ST_m^n + \hat{J}_j}{MUT_j} \right\rfloor + 1, & \text{if } \xi_j = M \end{cases} \quad (9)$$

Where,  $\varphi_j(\phi_k)$  is calculated by replacing the indices  $m$  and  $i$  with  $j$  and  $k$  in (6) respectively.  $\left\lfloor \frac{\hat{J}_j + \varphi_j(\phi_k)}{T_j} \right\rfloor$  represents the maximum number of instances of the higher priority periodic message or periodic copy of mixed message  $m_j$  that may accumulate at the critical instant. Whereas  $\left\lfloor \frac{ST_m^n - \varphi_j(\phi_k)}{T_j} \right\rfloor + 1$  represents the maximum number of instances of  $m_j$  that are queued for transmission in the interval that starts with the critical instance and ends at  $\Upsilon_m^n$ . It should be noted that we use  $\hat{J}_j$  that includes the additional jitter from a higher priority message. There are no offset relations of  $m_m$  with any sporadic message. Moreover, all sporadic messages are assumed to be queued

for transmission at the critical instant.  $\left\lfloor \frac{ST_m^n + \hat{J}_j}{MUT_j} \right\rfloor + 1$  represent the maximum number of instances of higher priority sporadic message or sporadic copy of mixed message  $m_j$  that are queued for transmission in the interval that starts with the critical instance and ends at  $\gamma_m^n$ . This also includes the number of instances of  $m_j$  that may accumulate at the critical instant due to jitter. It is evident from (9) that interference from both periodic and sporadic copies of every higher priority mixed message is taken into account. The lowest- and highest-numbered instances of  $m_m$  denoted by  $q_m^L$  and  $q_m^H$  are calculated as follows.

$$q_m^L = - \left\lfloor \frac{J_m + \varphi_m(\phi_i)}{T_m} \right\rfloor + 1 \quad (10)$$

$$q_m^H = \left\lfloor \frac{L_m - \varphi_m(\phi_i)}{T_m} \right\rfloor \quad (11)$$

Where  $L_m$  represents the length of priority level- $m$  busy period. We adapt the calculations for  $L_m$  from the existing analysis [28, 23] by including the additional jitter from higher priority messages and additional blocking delay.

$$L_m^{n+1} = \left[ \left\lfloor \frac{J_m + \varphi_m(\phi_i)}{T_m} \right\rfloor + \left\lfloor \frac{L_m^n - \varphi_m(\phi_i)}{T_m} \right\rfloor \right] \cdot C_m + \hat{B}_m + \sum_{\forall \Gamma_k \in \Gamma, m_j \in hp_k(m_m)} M_k^j(L_m^n) \cdot C_j \quad (12)$$

Where

$$M_k^j(L_m^n) = \begin{cases} \left\lfloor \frac{\hat{J}_j + \varphi_j(\phi_k)}{T_j} \right\rfloor + \left\lfloor \frac{L_m^n - \varphi_j(\phi_k)}{T_j} \right\rfloor, & \text{if } \xi_j = \text{P} \\ \left\lfloor \frac{L_m^n + \hat{J}_j}{MUT_j} \right\rfloor + 1, & \text{if } \xi_j = \text{S} \\ \left\lfloor \frac{\hat{J}_j + \varphi_j(\phi_k)}{T_j} \right\rfloor + \left\lfloor \frac{L_m^n - \varphi_j(\phi_k)}{T_j} \right\rfloor \\ + \left\lfloor \frac{L_m^n + \hat{J}_j}{MUT_j} \right\rfloor + 1, & \text{if } \xi_j = \text{M} \end{cases} \quad (13)$$

**Case 1(b): When  $m_m$  is a sporadic message** Let  $m_m$  belongs to the transaction of its own denoted by  $\Gamma_i$ . The worst-case response time of  $m_m$  can be calculated similar to the periodic case with one exception. That is, sporadic message does not hold any offset relations with any other message in the system. Moreover, all sporadic messages including  $m_m$  are assumed to be queued for transmission at the critical instant. Intuitively,  $\phi_i$  will be equal to  $MUT_m$ , i.e., the latest arrival of  $m_m$  prior to critical instant will be  $MUT_m$  time units before the critical instant. Let us use  $O_m$  equal to zero, and  $MUT_m$  in place of both  $T_m$  and  $\phi_i$  in (6).

$$\varphi_m(\phi_i) = 0 \quad (14)$$

In this case, (4), (7), (28), (8), (9) and (13) hold intact. However, we need to replace the new value of  $\varphi_m(\phi_i)$  from (14) in the calculations for (5), (10), (11) and (12) as follows. Moreover, we need to consider the effect of message copy time in its response time.

$$R_m(q_m) = ST_m + C_m + CT_m - (q_m - 1).MUT_m \quad (15)$$

$$q_m^L = -\left\lfloor \frac{J_m}{MUT_m} \right\rfloor + 1 \quad (16)$$

$$q_m^H = \left\lceil \frac{L_m}{MUT_m} \right\rceil \quad (17)$$

$$L_m^{n+1} = \left[ \left\lfloor \frac{J_m}{MUT_m} \right\rfloor + \left\lceil \frac{L_m^n}{MUT_m} \right\rceil \right] \cdot C_m + \hat{B}_m + \sum_{\forall \Gamma_k \in \Gamma, m_j \in hp_k(m_m)} \mathbf{M}_k^j(L_m^n) \cdot C_j \quad (18)$$

**Case 1(c): When  $m_m$  is a mixed message** Since a mixed message is duplicated as two separate messages, the extended analysis treats them separately. Let the periodic and sporadic copies of  $m_m$  be denoted by  $m_{m_P}$  and  $m_{m_S}$  respectively. We denote the worst-case response times of  $m_{m_P}$  and  $m_{m_S}$  by  $R_{m_P}$  and  $R_{m_S}$  respectively. The worst-case response time of  $m_m$  is the maximum between  $R_{m_P}$  and  $R_{m_S}$  as follows.

$$R_m = \max\{R_{m_P}, R_{m_S}\} \quad (19)$$

Where  $R_{m_P}$  and  $R_{m_S}$  are equal to the maximum value among the response times of their respective instances. Let  $q_{m_P}$  be the index variable to denote the instances of  $m_{m_P}$ . Let  $q_{m_P}^L$  and  $q_{m_P}^H$  denote the lowest- and highest-numbered instances of  $m_{m_P}$  respectively. Let  $q_{m_S}$ ,  $q_{m_S}^L$  and  $q_{m_S}^H$  denote the index variable for instances, and lowest- and highest-numbered instances of  $m_{m_S}$  respectively. The calculations for  $R_{m_P}$  and  $R_{m_S}$  are adapted from the periodic and sporadic cases respectively as follows.

$$R_{m_P} = \max\{R_{m_P}(q_{m_P})\}, \forall q_{m_P}^L \leq q_{m_P} \leq q_{m_P}^H \quad (20)$$

$$R_{m_S} = \max\{R_{m_S}(q_{m_S})\}, \forall q_{m_S}^L \leq q_{m_S} \leq q_{m_S}^H \quad (21)$$

The calculations for worst-case response time of each instance of  $m_{m_P}$  and  $m_{m_S}$  are adapted from (5) and (15) as follows.

$$R_{m_P}(q_{m_P}) = ST_{m_P} + C_m + CT_m - (\varphi_{m_P}(\phi_i) + (q_{m_P} - 1).T_m) \quad (22)$$

$$R_{m_S}(q_{m_S}) = ST_{m_S} + C_m + CT_m - (q_{m_S} - 1).MUT_m \quad (23)$$

Where  $\varphi_{m_P}(\phi_i)$  is calculated using (6). The calculations for  $ST_{m_P}$  and  $ST_{m_S}$  are adapted from (7) after some augmentation and adaptation of message copy times and additional jitter.

$$ST_{m_P}^{n+1} = \hat{B}_m + (q_{m_P} - q_{m_P}^L) \cdot C_m + (q_{m_P} - q_{m_P}^L) \cdot CT_m + Q_{m_S}^P \cdot C_m + \sum_{\forall \Gamma_k \in \Gamma} W_{m_P}(\Gamma_k, \phi_k, ST_{m_P}^n) \quad (24)$$

$$ST_{m_S}^{n+1} = \hat{B}_m + (q_{m_S} - q_{m_S}^L) \cdot C_m + (q_{m_S} - q_{m_S}^L) \cdot CT_m + Q_{m_P}^S \cdot C_m + \sum_{\forall \Gamma_k \in \Gamma} W_{m_S}(\Gamma_k, \phi_k, ST_{m_S}^n) \quad (25)$$

Where  $Q_{m_S}^P \cdot C_m$  and  $Q_{m_P}^S \cdot C_m$  in the above equations represent the effect of self interference in a mixed message. By self interference we mean that the periodic copy of a mixed message can be interfered by the sporadic copy and vice versa. Since, both  $m_{m_P}$  and  $m_{m_S}$  have equal priorities, any instance of  $m_{m_S}$  queued ahead of  $m_{m_P}$  will contribute an extra delay to the worst-case queueing delay experienced by  $m_{m_P}$ . A similar argument holds in the case of  $m_{m_S}$ . We adapt the calculations for the effect of self interference in a mixed message that we derived in [28]. It should be noted that the calculations for self interference differs from [28] in a way that we also consider a special case when both jitter and offset of a mixed message are zero. In this case, the previous calculations [28] result in zero self interference for the zeroth instances of  $m_{m_P}$  and  $m_{m_S}$ . However, in reality, even if  $J_m$  and  $O_m$  are zero, the zeroth instance of  $m_{m_P}$  can be interfered by one instance of  $m_{m_S}$  and vice versa. For example, consider  $m_m$  to be the highest priority message. Let  $m_{m_S}(0)$  is queued just after the queueing of  $m_{m_P}(0)$ . The instance  $m_{m_P}(0)$  can be blocked by any lower priority message. However,  $m_{m_S}(0)$  cannot start its transmission unless  $m_{m_P}(0)$  is transmitted. Therefore, we have to consider this specific case for the calculation of self interference as given below.

$$Q_{m_S}^P = \begin{cases} \left\lceil \frac{q_{m_P} \cdot T_m + J_m + O_m + \tau_{bit}}{MUT_m} \right\rceil, & \text{if all } \{q_{m_P}, J_m, O_m\} = 0 \\ \left\lceil \frac{q_{m_P} \cdot T_m + J_m + O_m}{MUT_m} \right\rceil, & \text{otherwise} \end{cases} \quad (26)$$

$$Q_{m_P}^S = \begin{cases} \left\lceil \frac{q_{m_S} \cdot MUT_m + J_m + O_m + \tau_{bit}}{T_m} \right\rceil, & \text{if all } \{q_{m_S}, J_m, O_m\} = 0 \\ \left\lceil \frac{q_{m_S} \cdot MUT_m + J_m + O_m}{T_m} \right\rceil, & \text{otherwise} \end{cases} \quad (27)$$

The calculations for  $W_{m_P}$ ,  $q_{m_P}^L$ ,  $q_{m_P}^H$  and  $L_{m_P}$  are done using (8), (10), (11) and (12) by replacing the index  $m$  with  $m_P$  respectively. Similarly,  $W_{m_S}$ ,  $q_{m_S}^L$ ,  $q_{m_S}^H$  and  $L_{m_S}$  are calculated using (8), (16), (17) and (18) by replacing the index  $m$  with  $m_S$  respectively. Further, the calculations in (3), (9) and (13) hold intact with proper replacement of the index variable for both  $m_{m_P}$  and  $m_{m_S}$ .

## 5.2 Case 2: When message under analysis is free from priority inversion

In this case, we consider that  $m_m$  is free from priority inversion because it belongs to the set of messages that contains the number of lowest priority messages equal to the number of transmit buffers in the node  $CC_i$ . In all three sub-cases corresponding to the periodic, sporadic and mixed message under analysis, most of the equations to calculate response time of  $m_m$  from Subsections 5.1, 5.1 and 5.1 are applicable respectively. However, the only exception lies in the calculations for start time and length of priority level- $m$  busy period in all three sub-cases. Since, the message under analysis is free from priority inversion, it will not experience additional blocking delay as shown in (3). On the other hand, the message under analysis does experience the additional jitter from higher priority messages. Moreover, copying delay for every instance of  $m_m$  in the priority level- $m$  busy period should also be considered. For this purpose, we replace the additional blocking delay  $\hat{B}_m$  by the original blocking delay  $B_m$  in (7), (12), (18), (24) and (25).  $B_m$  is defined as the amount of time equal to the largest transmission time in the set of lower priority messages and is calculated as follows.

$$B_m = \max_{\forall m_j \in lp(m_m)} \{C_j\} \quad (28)$$

## 6 Conclusion

The existing response-time analysis for CAN has limitations such that it does not support mixed messages that are scheduled with offsets in the systems where the CAN controllers implement abortable transmit buffers. Mixed messages are partly periodic and partly sporadic; and are implemented by several higher-level protocols for CAN that are used in the automotive industry today. We extended the existing analysis which is now applicable to any higher-level protocol for CAN that uses periodic, sporadic, and mixed transmission of messages that (only periodic and mixed messages) may be scheduled with offsets in the systems where the CAN controllers implement abortable transmit buffers. The extended analysis is also applicable to gateway nodes where jitter and deadlines of messages can be higher than their periods.

## Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) within the projects Synth-Soft and FEMMVA. The authors thank the industrial partners Arcticus Systems, BAE Systems Hägglunds and Volvo CE, Sweden.

## References

1. Robert Bosch GmbH, "CAN Specification Version 2.0," postfach 30 02 40, D-70442 Stuttgart, 1991.
2. "Automotive networks. CAN in Automation (CiA)," <http://www.can-cia.org/index.php?id=416>.
3. ISO 11898-1, "Road Vehicles interchange of digital information controller area network (CAN) for high-speed communication, ISO Standard-11898, Nov. 1993."



4. N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, 1993.
5. N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historic perspective," *Real-Time Systems*, vol. 8, no. 2/3, pp. 173–198, 1995.
6. L. Sha, T. Abdelzaher, K.-E. A. rzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, vol. 28, no. 2/3, pp. 101–155, 2004.
7. M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.
8. 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.
9. "Volcano Network Architect (VNA). Mentor Graphics," <http://www.mentor.com/products/vnd/communication-management/vna>.
10. 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.
11. Rubus-ICE, <http://www.arcticus-systems.com>.
12. S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Computer Science and Information Systems, ISSN: 1361-1384*, vol. 10, no. 1, 2013.
13. R. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Schedulability analysis for controller area network (can) with fifo queues priority queues and gateways," *Real-Time Systems*, vol. 49, no. 1, 2013.
14. O. Pfeiffer, A. Ayre, and C. Keydel, *Embedded Networking with CAN and CANopen*. Anna-books, 2003.
15. Marco Di Natale and Haibo Zeng, "Practical issues with the timing analysis of the Controller Area Network," in *18th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2013.
16. 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.
17. 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.
18. 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.
19. A. Szakaly, "Response Time Analysis with Offsets for CAN," Master's thesis, Department of Computer Engineering, Chalmers University of Technology, Nov. 2003.
20. Y. Chen, R. Kurachi, H. Takada, and G. Zeng, "Schedulability comparison for can message with offset: Priority queue versus fifo queue," in *19th International Conference on Real-Time and Network Systems (RTNS)*, Sep. 2011, pp. 181–192.
21. L. Du and G. Xu, "Worst case response time analysis for can messages with offsets," in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, nov. 2009, pp. 41–45.
22. R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.
23. P. Yomsi, D. Bertrand, N. Navet, and R. Davis, "Controller Area Network (CAN): Response Time Analysis with Offsets," in *9th IEEE International Workshop on Factory Communication Systems*, May 2012.

24. 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.
25. 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.
26. 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.
27. 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.
28. S. Mubeen, J. Mäki-Turja and M. Sjödin, "Extending Offset-Based Response-Time Analysis for Mixed Messages in Controller Area Network," in *18th IEEE Conference on Emerging Technologies and Factory Automation*, sept. 2013.
29. CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Ver.4.02. Feb., 2002.
30. Requirements on Communication, Release 3.0, Rev 7, Ver. 2.2.0. The AUTOSAR Consortium, Sep., 2010.
31. "AUTOSAR Technical Overview, Version 2.2.2., Release 3.1, The AUTOSAR Consortium, Aug., 2008," <http://autosar.org>.
32. "Häggglunds Controller Area Network (HCAN), Network Implementation Specification," *BAE Systems Häggglunds, Sweden (internal document)*, April 2009.
33. J. Palencia and M. G. Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets," *Real-Time Systems Symposium*, 1998.