Worst-Case Response-Time Analysis for Mixed Messages with Offsets in Controller Area Network[*]

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 Controller Area Network (CAN) does not support mixed messages that are scheduled with offsets. Mixed messages are implemented by several high-level protocols for CAN that are used in the automotive industry. We extend the existing offset-based analysis which is applicable to any high-level protocol for CAN that uses periodic, sporadic and mixed transmission of messages. Moreover, we implement the extended analysis as a standalone simulator that will be integrated as a plug-in with the existing industrial tool suite (Rubus-ICE). The experiments, that we performed, indicate that it is possible to achieve up to 4.48% improvement in schedulability when mixed messages are scheduled with offsets.

1. Introduction

The Controller Area Network (CAN) [24] is one of the widely used real-time networks in automotive domain. It is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 mega bits per second. It has been standardized by the International Organization for Standardization as ISO 11898-1 [17]. According to CAN in Automation (CiA) [2], the estimated number of CAN enabled controllers sold in 2011 are about 850 million. CAN also finds its applications in other domains such as industrial control, medical equipments, maritime electronics, production machinery, etc. 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).

System providers of hard real-time systems are required to ensure that the system meets its deadlines. In order to provide evidence that each action by the system will be provided in a timely manner, i.e., each action will be taken at a time that is appropriate to the environment of the system, *a priori* analysis techniques, such as schedulability analysis, have been developed by the research community. Response-Time Analysis (RTA) [8, 26] is a powerful, mature and well established schedulability analysis technique. It is a method to calculate upper bounds on the response times of tasks or messages in a real-time system or a real-time network respectively. In crux, RTA is used to perform a schedulability test which means it checks whether or not tasks (or messages) in the system (or network) will satisfy their deadlines. RTA applies to systems (or networks) where tasks (or messages) are scheduled with respect to their priorities and which is the predominant scheduling technique used in real-time operating systems (or real-time network protocols, e.g., CAN) today [23].

1.1. Motivation and related work

The schedulability analysis of CAN was developed by Tindell et al. [29] by adapting the theory of fixed priority preemptive scheduling for uniprocessor systems. This analysis has been implemented in the analysis tools that are used in the automotive industry [6, 21, 22]. The analysis has also served as the basis for many research projects. Later on, Davis et al. [11] refuted, revisited and revised the analysis developed by Tindell et al. In [12], Davis et al. extended the analysis in [29, 11] which is applicable to the CAN network where some nodes implement priority queues and some implement FIFO queues. All these analyses assume that the messages are queued for transmission periodically or sporadically. They do not support mixed messages in CAN, i.e., the messages that are simultaneously time (periodic) and event (sporadic) triggered.

Mubeen et al. [20] extended the existing analysis to support the worst-case response time computation of mixed messages in CAN where the nodes implement priority-based queues. Recently, Mubeen et al. [25, 19] further extended the analysis for CAN to support mixed messages in the network where some nodes implement priority-based queues while others implement FIFO-based 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.

In order to avoid deadlines violations due to high transient loads, current automotive embedded systems are often scheduled with offsets [30]. Furthermore, the worst-case response times of messages (especially with lower pri-

[*]test

ority) in CAN increase with the increase in the network load. However, the worst-case response-times of lower priority messages in CAN can be reduced if the messages are scheduled with offsets [27, 14, 15]. A method for the assignment of offsets to improve the overall bandwidth utilization is proposed in [14, 15]. The worst-case response-time analysis for CAN messages with offsets has been developed by several researchers [10, 13, 27, 30].

However, none of the existing offset-based analysis for CAN facilitates the response-time computation of mixed messages. Hence, an extension in the existing offset-based analysis for CAN messages is required to support mixed messages. Among several existing analyses for CAN messages with offsets, we selected to extend the analysis in [10] for mixed messages because it provides sufficient and safe upper bounds on the response times, has lower computational complexity and supports the analysis of CAN messages in the priority-queued as well as FIFO-queued nodes.

1.2. Paper contribution

We identified that the existing offset-based RTA for CAN [10, 13, 27, 16] does not support the analysis of common message transmission patterns (i.e., mixed messages) which are implemented by some high-level protocols used in the industry. We extended the existing offset-based RTA for CAN [10] to support the worst-case response-time computation of mixed messages. The extended analysis is applicable to any high-level protocol for CAN that uses periodic, sporadic and mixed transmission of messages that are scheduled with offsets. We implemented the extended analysis as a standalone simulator that will be integrated as a plug-in with the existing industrial tool suite Rubus-ICE [1, 21]. We also performed a number of experiments by analyzing practical sets of messages with both the existing analysis of mixed messages without offsets [20] and the extended analysis of mixed messages with offsets. We observed up to 4.48% improvement in system schedulability when messages are scheduled with offsets.

1.3. Paper layout

The rest of the paper is organized as follows. In Section 2, we discuss mixed transmission patterns supported by several high-level protocols for CAN. Section 3 describes the scheduling model. In Section 4, we extend the existing analysis. In Section 5, we discuss the implementation and evaluation of the analysis. Section 6 concludes the paper and discusses the future work.

2. Mixed transmission patterns supported by high-level protocols

When CAN is employed for network communication in a distributed real-time system, each node (processor) is equipped with a CAN interface that connects the node to the bus [28]. Application tasks in each node, that require remote transmission, are assumed to queue messages for transmission over CAN bus. The messages are actually transmitted according to the protocol specification of CAN. The existing analyses for CAN messages with offsets [10, 13, 27, 16] assumes that the tasks queueing CAN messages are invoked either by periodic events with a period or sporadic events with a minimum inter-arrival time. However, there are some high-level protocols and commercial extensions of CAN in which the task that queues the messages can be invoked periodically as well as sporadically. If a message can be queued for transmission periodically as well as at the arrival of a sporadic event then the transmission type of the message is said to be mixed. In other words, a mixed message is simultaneously time (periodic) and event triggered (sporadic). We identified three types of implementations of mixed messages used in the industry.

Consistent terminology. To stay consistent, we will use the terms message and frame interchangeably because we only consider messages that will fit into one frame (maximum 8 bytes). For the purpose of using simple notation, we will call a CAN message as periodic, sporadic or mixed if it is queued by an application task that is invoked periodically, sporadically or both (periodically and sporadically) respectively. If a message is queued for transmission at periodic intervals, we will 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 a Minimum Update Time (MUT) between the queueing of two successive sporadic messages has elapsed. Hence, the transmission of a sporadic frame is constrained by MUT. We will overload the term "MUT" to refer to "Inhibit Time" in CANopen protocol [4] and "Minimum Delay Time (MDT)" in AUTOSAR communication [5].

2.1. Method 1: Implementation of a mixed message by CANopen

The CANopen protocol [4] 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 [4].

The transmission pattern of a mixed message in CANopen is illustrated in Figure 1. The down-pointing arrows symbolize the queueing of messages while the upward lines (labeled with alphabets) 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. Similarly, message 3 is queued due to the expiry of the Event Timer. When the 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 Figure 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.

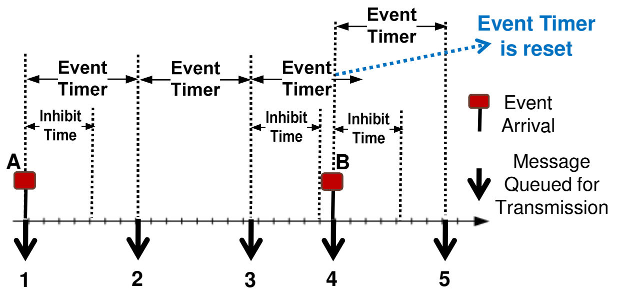

Figure 1. Mixed transmission pattern in CANopen

2.2. Method 2: Implementation of a mixed message by AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) [3] 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 Figure 2. Message 1 is queued (MDT is started) because of partly periodic nature of a mixed message. When the 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 Figure 2. The periodic transmissions corresponding to messages 5 and 6 take place at the scheduled time because MDT is already expired in both cases.

2.3. Method 3: Implementation of a mixed message by HCAN

A mixed message defined by HCAN protocol [7] 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

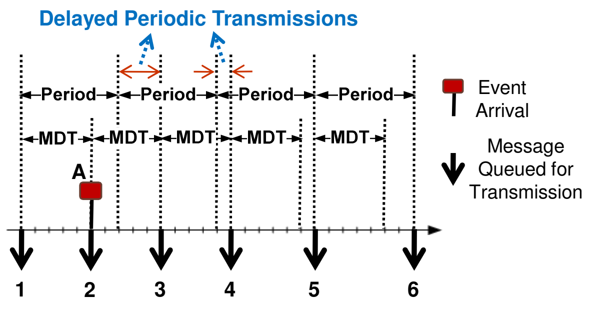
Delayed Periodic Transmissions
Period
Period
Period
Period
MDT
MDT
MDT
MDT
MDT
A
Event Arrival
Message Queued for Transmission
1
2
3
4
5
6

Figure 2. Mixed transmission pattern in AUTOSAR

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 illustrated in Figure 3.

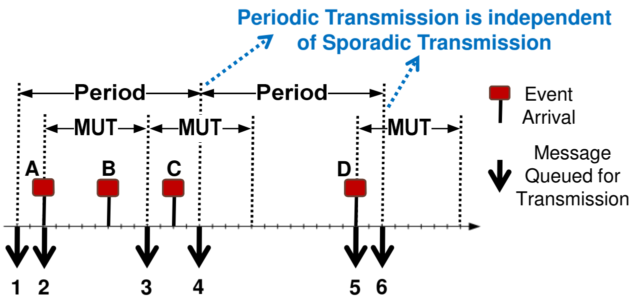
Periodic Transmission is independent of Sporadic Transmission
Period
Period
MUT
MUT
MUT
A
B
C
D
Event Arrival
Message Queued for Transmission
1
2
3
4
5
6

Figure 3. Mixed transmission pattern in HCAN

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 its sporadic transmission (a unique property of HCAN protocol). When the 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 the periodicity.

2.4. Discussion

In the first method, the Event Timer is reset every time a mixed message is queued for transmission. The implementation of a 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 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 for CAN messages with offsets [10, 13, 27, 16, 30] can be used for analyzing mixed messages in the first and second implementation methods.

However, the periodic transmission is independent of the sporadic transmission in the third method. The periodic timer is not reset with every sporadic transmission. 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 analyses in [10, 13, 27, 16, 30] cannot be used for analyzing mixed messages in the third implementation method. This calls for the need to extend the existing analysis of CAN messages with offsets to support mixed messages.

3. System scheduling model

The system scheduling model extends the scheduling model for the response-time analysis of mixed messages in CAN [20] by adding offset-related information from the system model of CAN messages with offsets [10]. The system, S, 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, i.e., the highest priority message in a node enters into the bus arbitration. 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.

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. Although the priorities of CAN messages are unique, the set $hep(m)$ will be used in the case of mixed messages. Associated to each message is a $FRAME\_TYPE$ that specifies whether the frame is a standard or an extended CAN frame. The difference between the two frame types is that the standard CAN frame uses an 11-bit identifier whereas the extended CAN frame uses a 29-bit identifier. In order to keep the notations simple and consistent, we define a function $\xi(m)$ that denotes the transmission type of a message. $\xi(m)$ specifies whether m 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 m has a transmission time (C_m) and queueing jitter (J_m) which is inherited from the task that queues m, i.e., the sending task. Each message can carry a data payload that ranges from 0 to 8 bytes. This number is specified in a header field of the frame called Data Length Code and denoted by s_m. In the case of periodic transmission, m has a period which is denoted by T_m. Whereas in the case of sporadic transmission, m has a MUT_m that refers to the minimum time that should elapse between the transmission of any two sporadic messages. B_m denotes the blocking time of m which refers to the largest amount of time m can be blocked by any lower priority message.

We duplicate a message when its transmission type is mixed. Hence, each mixed message m is treated 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. 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). A message m is deemed schedulable if its R_m is less than or equal to its deadline D_m. A system S is considered schedulable if all of its messages are schedulable. We assume the deadline of a message is less than or equal to its period or MUT.

Let O_m denotes the offset of m. We assume that the offset of m is always smaller than its period or minimum update time. Let the arrival times of m be denoted by $A_m^n(where, n = 0, 1, 2, ...)$. The first arrival time of m is equal to its offset, i.e., $A_m^0 = O_m$. The subsequent arrivals of m occur periodically with respect to its first arrival. It should be noted that there is no global synchronization among CAN controllers. Therefore, A_m^n is the local time of a CAN controller that transmits m. We assume that the offset relations exist only among the periodic messages and periodic copies of mixed messages within a node. We further assume that 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 the duplicates of a mixed message.
5. Between any two periodic messages belonging to different nodes.

All periodic messages and periodic copies of mixed messages in a node (say) CC_c belong to a single transaction denoted by Γ_c^P. All sporadic messages in node CC_c are combined in a single transaction denoted by Γ_c^S. There is no relation among the messages in this transaction except that all of them are sporadic and belong to the same node. Similarly, sporadic copies of all mixed messages in node CC_c are combined in a single transaction denoted by $\Gamma_c^{M_S}$. It should be emphasized again that the offset relations exist only within Γ_c^P.

We define an operator $\bigoplus$ that adds multiple functions with maximum slope equal to 1. It will be used to add multiple interferences in the extended analysis (next Section). $\bigoplus$ operation is demonstrated in Figure 4. There are two functions of time , i.e., $f(t)$ and $g(t)$ whose graphs are depicted in Figure 4(a) and 4(b) respectively. By definition, $\bigoplus$ operator adds two functions such that their sum at any value of time (say) t_1 cannot be greater than t_1. If sum of the functions (at $t = t_1$) is greater than t_1 then $f(t_1) \bigoplus g(t_1)$ will be assigned the value that is equal to t_1. For example, at t equal to 1 the sum of $f(1)$ and $g(1)$ is 2 (i.e., $1 + 1$). However by definition, the value of $f(1) \bigoplus g(1)$ cannot be greater than 1 (i.e., the current value of t). Hence, $f(1) \bigoplus g(1)$ is assigned the current value of t, i.e., 1 as shown in Figure 4(c). Similarly, at t equal to 3 the sum of $f(3)$ and $g(3)$ is 4 (i.e., $2 + 2$) which is greater than the current value of t (i.e., 3). Therefore, $f(3) \bigoplus g(3)$ is assigned the current value of t, i.e., 3. Now, consider the time when t is equal to 5. The value of the sum of $f(5)$ and $g(5)$ is 5 (i.e., $3 + 2$) which is not greater than the current value of t (i.e., 5). Therefore, $f(5) \bigoplus g(5)$ is equal to 5.

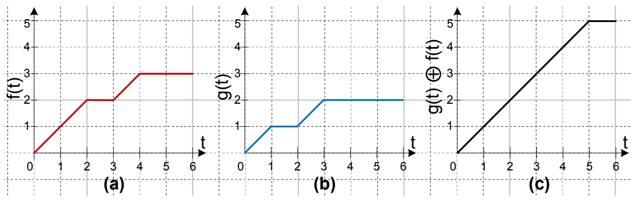

Figure 4. Demonstration of $\bigoplus$ operation

4. Response-time analysis of mixed messages with offsets

We will treat a message differently based on its transmission type. Hence, we will consider three different cases. Let m be the message under analysis that belongs to the node CC_c.

4.1. Case: When *m* is a periodic message

In order to calculate the worst-case response time of m, the maximum busy period for priority level-m should be known first. The maximum busy period is a term defined in the classical response-time analysis [29, 11] as the longest contiguous interval of time during which m is unable to complete its transmission because (1) the bus is occupied by the higher priority messages in the system (2) a lower priority message already started its transmission when m is queued for transmission.

The maximum busy period starts at the so-called critical instant [18]. In a system where messages are scheduled without offsets, the critical instant is the point in time when all higher priority messages are assumed to be queued simultaneously with m while their subsequent instances are assumed to be queued after the shortest possible interval of time [11]. However, this assumption does not hold in the system where messages are scheduled with offsets. Let us denote the set of periodic messages and periodic copies of mixed messages in the node CC_j with priorities higher

CC	m	P_m	ξ_m	T_m	MUT_m	C_m	O_m
CC_1	m_1	1	M	10	10	1	2
CC_1	m_2	2	P	10	-	1	0
CC_1	m_3	3	P	10	-	1	4
CC_2	m_5	5	P	10	-	1	0
CC_2	m_6	6	P	10	-	1	1
CC_3	m_4	4	S	-	10	1	-
CC_3	m_7	7	P	10	-	1	0

Table 1. Example message set containing periodic, sporadic and mixed messages with offsets

than m by $hp_j^{P,M_P}(m)$. Similarly, $hp_j^{S,M_S}(m)$ denotes the set of sporadic messages and sporadic copies of mixed messages in the node CC_j with priorities higher than m. We redefine the critical instant as the instant that meets all the following conditions.

1. *Condition 1:* m or any other higher priority message belonging to node CC_c is queued for transmission.
2. *Condition 2:* At least one message from the set $hp_j^{P,M_P}(m)$ from every node CC_j, other than node CC_c, is queued at its respective node.
3. *Condition 3:* All messages in the set $hp_j^{S,M_S}(m)$ from every node CC_j, including node CC_c, are simultaneously queued at their respective nodes.
4. *Condition 4:* A lower priority message just started its transmission when m is queued.

Worst-Case Candidates

All of the above conditions, except condition 2, are easy to check. The problem here is that which message in the set $hp_j^{P,M_P}(m)$ in each node is the candidate to start the critical instant and hence, contributes to the worst-case queueing delay [11] for m. The answer is that any message in this set can be the worst-case candidate. Hence, we need to check each message from this set in every node in the interval $[0, LCM]$ as the potential worst-case candidate. LCM is the least common multiple of the periods of all messages in the set $hp_j^{P,M_P}(m)$ in the corresponding node CC_j. It should be noted that sporadic messages and sporadic copies of mixed messages are not considered while calculating LCM. The arrival of all these messages are repeated periodically after every LCM time in their respective node. The response time of m should be computed from every worst-case candidate and the maximum among all should be considered as the worst-case response time of m.

Consider an example system consisting of three nodes as shown in Table 1. There are seven messages that are sent by these nodes. The timing attributes are specified in $msec$. The queueing jitter of all the messages in Table 1 is assumed to be zero. Let the message under analysis be

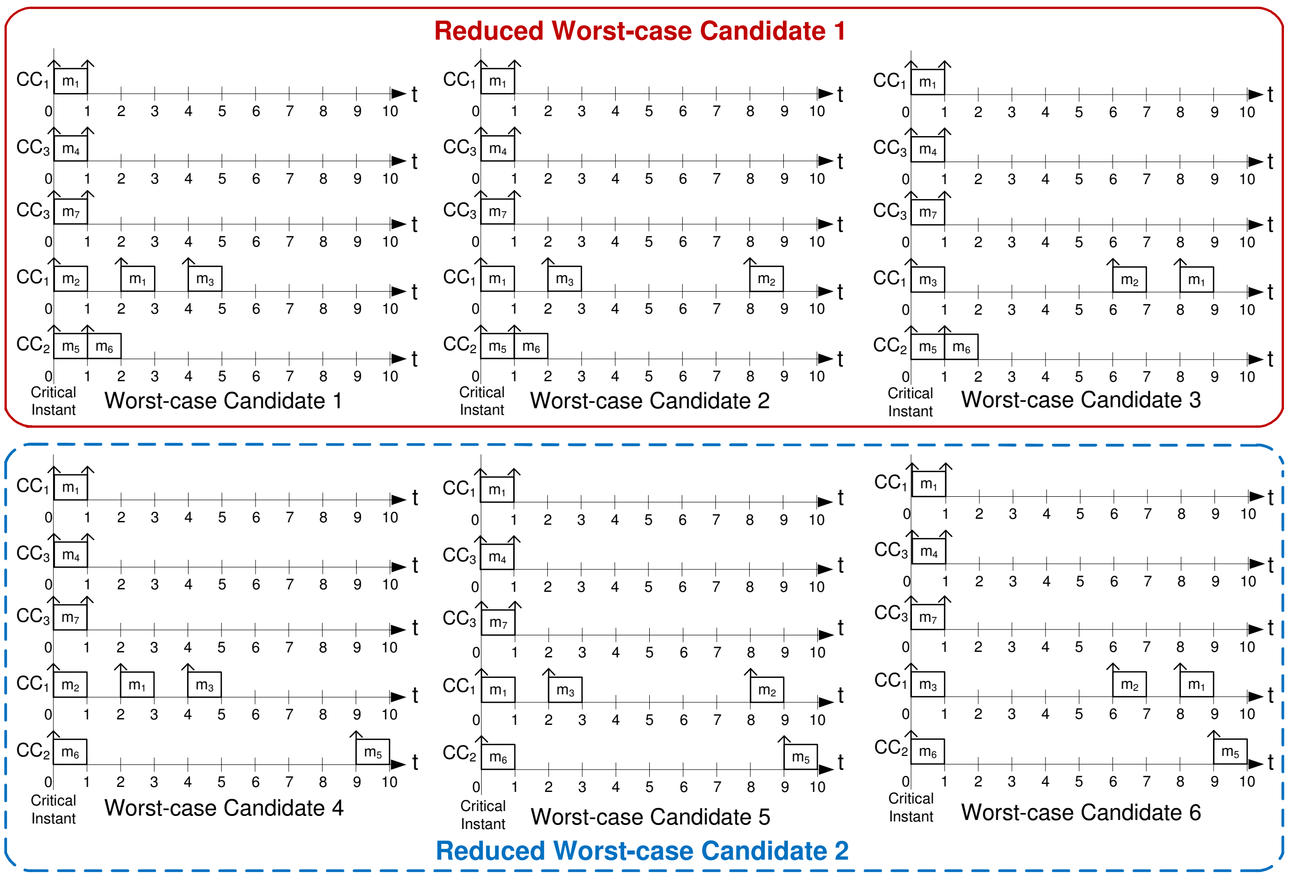

Figure 5. Worst-case candidates for m$_6$ in the example message set shown in Table 1

m_6 that belongs to node CC_2. There are six scenarios that depict the worst-case candidates that meet the four conditions as shown in Figure 5. In all six scenarios: the first and second time lines (from the top) correspond to condition 3; third time line corresponds to condition 4; fourth time line corresponds to condition 2; and the last time line corresponds to condition 1.

Factors contributing to the response-time of *m*

There are six factors that contribute to the worst-case response time of m.

1) Blocking delay. If a lower priority message just starts its transmission when m is queued for transmission then m has to wait in the send queue and is said to be blocked by it. The lower priority message cannot be preempted once it starts its transmission as CAN uses fixed-priority non-preemptive scheduling. m can also be blocked from its own previous instance due to push-through blocking [11]. The maximum blocking delay for m is denoted by B_m. It can be calculated as follows.

$$B_m = \max_{\forall k \in \aleph \wedge P_k \geq P_m} C_k \qquad (1)$$

2) Delay due to interference from the message sets *hp*^{P,M_P}*(m)* belonging to all nodes except *CC*$_c$. Since CAN uses fixed-priority non-preemptive scheduling, a message cannot be interfered by higher priority messages during its transmission on the bus. Whenever we use the term interference, it refers to the amount of time m has to wait in the send queue because the higher priority messages win the arbitration, i.e., the right to transmit before m.

Interference Function (IF). The definition of IF is adapted from [10]. Formally, $IF_c^{ST}[P_m](t)$ represents the interference in a time interval $[ST, t]$ received by m from the message set $hp^{P,M_P}(m)$ in node CC_c. ST represents the starting time of the interference function. It should be noted that there is a limitation in the existing analysis [10] that IF considers only periodic higher priority messages.

Consider again the example system shown in Table 1. The node CC_1 sends three messages, i.e., m_1, m_2 and m_3. The least common multiple of their periods, denoted by LCM_1, is equal to 10 as shown in Figure 6. There are three arrival times of these messages in the interval $[0, LCM_1]$, i.e, 0, 2, and 4 as shown in Figure 6(a). Hence, there will be three IF functions (corresponding to these three arrival times) that may cause interference to a lower priority message (say m_6). These three interference functions denoted by $IF_1^0[P_6](t)$, $IF_1^2[P_6](t)$ and $IF_1^4[P_6](t)$ are shown in Figure 6(a), 6(b) and 6(c) respectively. Although, we assumed the queueing jitter of the messages to be zero in this example, the effect of jitter from every message should be taken into account when IF function is drawn.

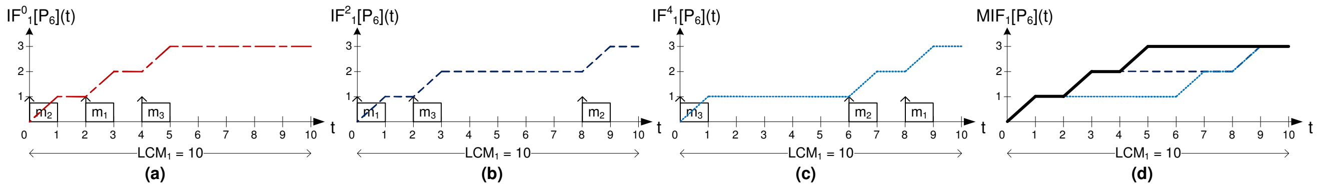

Figure 6. Graphs of IFs and MIF in CC_1 in the example message set shown in Table 1

For the example system shown in Table 1, there are six worst-case candidates. In practical systems, there can be hundreds of messages and the set of worst-case candidates may become too large and hence, the computational complexity of the analysis may become too high. In order to reduce this complexity, all IF functions in a node may be replaced by a single function MIF that is adapted from the existing analysis [10].

Maximum Interference Function (MIF). The maximum interference function, denoted by $MIF_j[P_m](t)$, is defined as the maximum interference received by m in an interval from the set $hp^{P,M_P}(m)$ belonging to the node CC_j. It should be noted that MIF in the existing analysis [10] considers only periodic higher priority messages. MIF can be calculated as follows.

$$MIF_j[P_m](t) = \max_{k \in hp_j^{P,M_P}(m), 0 \le A_k^n \le LCM_j} IF_j^{A_k^n}[P_m](t) \tag{2}$$

Where, LCM_j refers to the least common multiple of periods of all messages in the set $hp_j^{P,M_P}(m)$. Consider again the message set belonging to node CC_1 as shown in Table 1. The maximum interference function for a lower priority message (say m_6) received from node CC_1, denoted by $MIF_1[P_6](t)$, will be the maximum of the three interference functions $IF_1^0[P_6](t)$, $IF_1^2[P_6](t)$ and $IF_1^4[P_6](t)$ at each instant as shown by the bold line graph in Figure 6(d).

The worst-case candidates for m_6 are reduced from 6 to 2 by using MIF instead of individual IFs. That is, first three candidates can be combined by replacing their fourth time line (from the top) by a single function $MIF_1[P_6](t)$. We call this combined candidate as the reduced worst-case candidate 1 as shown in the upper block in Figure 5. It should be noted that the computational complexity of the analysis is reduced by replacing a single MIF function instead of several IFs from the same node but the analysis may not remain exact, although, it is sufficient and safe [10].

3) Delay due to interference from sporadic messages from all nodes. We define a sporadic interference function to represent delay due to interference from sporadic messages from each node. The *Sporadic Interference Function* (IF_j^S) represents the interference received by m from a set of sporadic messages that belong to the node CC_j and have priorities higher than P_m. Mathematically, it can be represented as follows.

$$IF_j^S = \sum_{\forall i \in hp_j(m) \wedge i \in \Gamma_j^S}^{\oplus} I_i \tag{3}$$

Where, I_i represents the interference by a higher priority sporadic message i in the interval $[ST, t]$. It should be noted that I_i may contain more than one instances of i. The number of instances of i in the interval $[ST, t]$ is equal to $\lceil(\text{sizeof}[ST, t]/MUT_i)\rceil$. In (3), $\bigoplus$ operation adds multiple interferences with maximum slope equal to 1. Consider m_6 as the message under analysis in the example system of Table 1. The sporadic interference function from node CC_3 is shown as $IF_3^S[P_6](t)$ in Figure 7.

4) Delay due to interference from sporadic copies of mixed messages from all nodes. We define a mixed sporadic interference function to represent delay due to interference from sporadic copies of mixed messages from each node. The *Mixed Sporadic Interference Function* ($IF_j^{M_S}$) represents the interference received by m from a set of sporadic copies of mixed messages that belong to the node CC_j and have priorities higher than P_m. Mathematically it can be represented as follows.

$$IF_j^{SM} = \sum_{\forall i \in hep_j(m) \wedge i \in \Gamma_j^{M_S}}^{\oplus} I_i \tag{4}$$

Where, I_i represents the interference by the sporadic copy of a higher priority mixed message i in the interval $[ST, t]$. Once again, I_i may contain more than one instances of i. The number of instances of i in the interval $[ST, t]$ is equal to $\lceil(\text{sizeof}[ST, t]/MUT_i)\rceil$. It should be noted that the set $hep(m)$ is used in (4) as compared to (3) where $hp(m)$ was used. This is because we need to consider the effect of self interference when a message under analysis is mixed. That is, the periodic copy of a mixed message m can be interfered by its sporadic copy. Consider m_6 as the message under analysis in the example system of Table 1. The mixed sporadic interference function from node CC_1 is shown as $IF_1^{MS}[P_6](t)$ in Figure 7. Although, we assumed the queueing jitter of the messages to be zero in this example, the effect of jitter from every message should be taken into account when IF_j^S and $IF_j^{M_S}$ functions are drawn.

5) Delay due to interference from the set $hp^{P,M_P}(m)$ belonging to CC_c. All messages in the set $hp^{P,M_P}(m)$ belonging to the same node CC_c as that of m can add delay to the response time of m if they are queued in the maximum busy period. This interference is denoted by $IF_c^{ST}[P_m](t)$. ST belongs to a set WT_m that includes the arrival times of all candidate messages from the node CC_c within the interval $[0, LCM_c]$ [10]. Consider m_6 as the message under analysis in the example system shown in Table 1. The set WT_6 will contain the first arrival times of m_5 and m_6, i.e., $WT_6 = \{A_5^0, A_6^0\} = \{0, 1\}$. So, the possible values for ST are 0, and 1 that correspond to the two reduced worst-case candidates as shown in their last time lines in Figure 5.

6) Queueing jitter of *m*. It is equal to the difference between the worst-case and best-case response times of the task that queues m.

Calculations for the worst-case response-time of *m*

The response time of m is calculated for each value of ST by combining all six delays due to interferences, blocking and jitter as follows.

$$R_m^{ST} = EIT\Bigg[B_{ST} \oplus IF_c^{ST}[P_m](t) \oplus \sum_{\forall CC_j}^{\oplus} IF_j^S \oplus \sum_{\forall CC_j, CC_j \neq CC_c}^{\oplus} MIF_j[P_m](t) \oplus \sum_{\forall CC_j}^{\oplus} IF_j^{M_S}\Bigg] + C_m + J_m + ST - A_m^{ST} \quad (5)$$

Where, the operation $EIT(X)$ is adapted from the existing analysis [10]. Basically, it calculates the time at which the slope of the function X becomes zero, i.e., the earliest bus idle time to transmit the message under analysis. Consider again the example message set shown in Table 1. Let m_6 be the message under analysis. Figure 7 demonstrates the computation of response time of m_6 for the reduced worst case candidate 1 (that corresponds to $ST = 0$). All the factors contributing to the response-time of m_6 for $ST = 0$ are also shown in Figure 7. In this case, the value of $EIT(X)$ is equal to 7 time units. Similarly, the calculated response-time of m in this case is:

$$R_6^0 = EIT(X) + C_6 + J_6 + ST - A_6^{ST} = 7+1+0+0-1 = 7$$

The worst-case response time of m denoted by R_m is the largest value among the response times of m that are computed for all values of ST.

$$R_m = \max_{\forall ST \in \{WT_m\}} R_m^{ST} \quad (6)$$

4.2. Case: When *m* is a sporadic message

Let again the message under analysis be m that belongs to the node CC_c. Since a sporadic message does not have any offset relations with any other message in the system, we will not consider the interference due to the fifth delay factor (discussed in the previous subsection). Instead, we will consider the interference from the node CC_c as well in

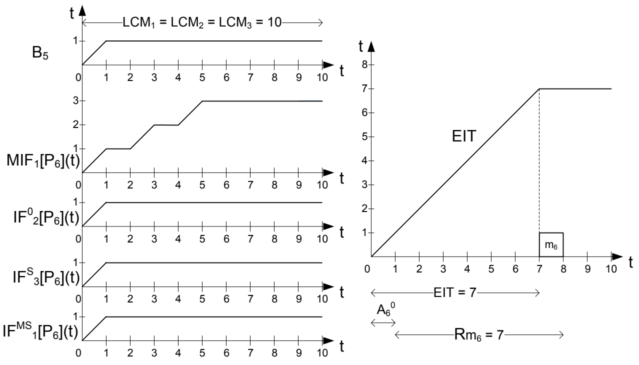

Figure 7. Demonstration for the calculation of response time of m$_6$ from its reduced worst-case candidate 1

the second delay factor (discussed in the previous subsection). The worst-case response time of a sporadic message m can be calculated as follows.

$$R_m = EIT\Bigg[B_m \oplus \sum_{\forall CC_j}^{\oplus} MIF_j[P_m](t) \oplus \sum_{\forall CC_j}^{\oplus} IF_j^S \oplus \sum_{\forall CC_j}^{\oplus} IF_j^{M_S}\Bigg] + C_m + J_m \quad (7)$$

The difference between (5) and (7) is that equation (7) does not include $IF_c^{ST}[P_m](t)$, ST and A_m^{ST}. Moreover, the calculation of MIF in (7) includes all nodes in the system.

4.3. Case: When *m* is a mixed message

Since a mixed message is duplicated as two separate messages, the extended analysis treats them separately. Let m be the message under analysis that belongs to the node CC_c. Let the periodic and sporadic copies of m be denoted by m_P and m_E. The worst-case response time of m_P, denoted by R_{m_P}, is computed in a similar fashion as that of a periodic message using (5) and (6). Similarly, the worst-case response time of m_E, denoted by R_{m_E}, is computed using (7). The worst-case response time of m is the maximum between R_{m_P} and R_{m_E} as follows.

$$R_m = max(R_{m_P},\ R_{m_E}) \quad (8)$$

5. Implementation and evaluation

The offset-aware analysis that we extended for mixed messages in the previous Section is also implemented as a standalone simulator. This simulator will be integrated as a plug-in with the existing industrial tool suite (Rubus-ICE) after extensive evaluation. Rubus-ICE is used for model- and component-based development of distributed real-time systems in automotive domain by several international companies. It supports several high-level protocols for CAN. There already exists a plug-in in Rubus-ICE

Message	m_1	m_2	m_3	m_4	m_5	m_6	m_7
R_m	2	4	5	6	7	8	8
R_m^{Offset}	2	3	6	5	6	7	8

Table 2. Calculated response times of message set shown in Table 1.

that implements the existing analysis for mixed messages in CAN that are scheduled without offsets [20, 21]. This plug-in is already in the industrial use.

Let us analyze the example message set in Table 1 with the existing and extended analysis for mixed messages. First, we analyze the messages without their offset information by using the existing analysis plug-in in Rubus-ICE. Second, we analyze the messages along with their offset-related information with the new standalone simulator that implements the extended analysis. The worst-case response times of the messages calculated in both the cases are listed in Table 2.

The analysis results indicate that the worst-case response times of most of the messages decrease if the messages in the example message set are scheduled with offsets. Only one message m_3 has a higher response time comparatively when scheduled with offsets. In Table 2, R_m^{Offset} and R_m represent the worst-case response times of messages calculated from the offset-aware analysis of mixed messages (extended in this paper) and the existing analysis of mixed messages that does not considers offsets [20] respectively.

We generated several message sets using the NETCARBENCH [9] tool that assigns offsets to messages according to the offset assignment method in [14, 15]. There is a limitation in NETCARBENCH that it cannot generate mixed messages. Therefore we had to modify the generated message sets manually by randomly selecting certain percentage of messages from the generated message sets as mixed, sporadic and periodic. Off course, the modified message set does not remain optimized according to the offset-assignment method in [14, 15].

Figure 8 shows the graph between R_m/D_m (ratio of the response times to the corresponding deadlines) and message priorities in one of the modified message sets generated from NETCARBENCH tool. The system consists of the following parameters: 10 Nodes; 50 messages; 250 Kbps CAN bus speed; 50% network load. Moreover, we selected 40%, 10% and 50% messages from the generated message set as mixed, sporadic and periodic respectively. The response times were calculated separately using the existing analysis plug-in in Rubus-ICE (that does not support offset-aware analysis for mixed messages) and the new simulator that implements offset-aware analysis for mixed messages. The deadlines of messages are the same in both the cases.

In this experiment, we observed that the worst-case response time of 68% messages decreased when the messages were scheduled with offsets. Whereas, the worst-case response times of 28% and 4% messages increased and remained the same when scheduled with offsets respectively. We observed 4.48% improvement in the schedulability of the system when the messages are scheduled with offsets. It should be noted that the minimum improvement in schedulability in all experiments was 1.12%. The percentage improvement in schedulability is calculated as follows:

$$\left[\left(\sum_{\forall m \in \aleph}\left[\frac{R_m - R_m^{Offset}}{D_m}\right]\right) \Big/ (sizeof(\aleph))\right] * 100$$

Where, $\aleph$ represents the set of all messages in the system.

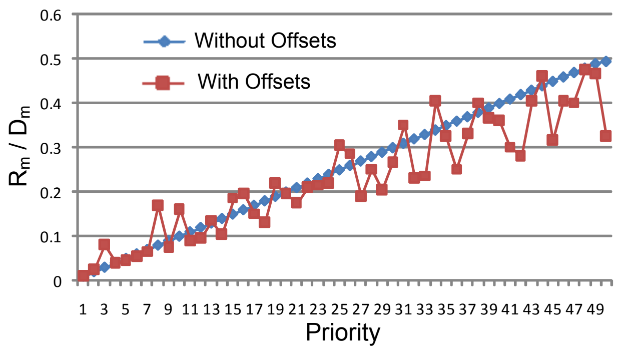

Figure 8. Plot of (R_m/D_m) against priorities for messages scheduled with and without offsets

It is interesting to note from Figure 8 that the offset-aware analysis sometimes calculates higher worst-case response times as compared to that of the analysis without offsets. The reason behind this anomaly is that the message sets in our experiments were generated from the NETCARBENCH tool that implements the offset-assignment method that is optimized for only periodic messages [14, 15]. Since NETCARBENCH does not support mixed messages, we manually introduced a certain percentage of mixed messages (discussed above) within the message sets generated from NETCARBENCH. Therefore, the offset assignment does not remain optimized for the modified message sets.

We believe, this anomalous effect can be significantly minimized while the percentage improvement in schedulability can be maximized by developing an optimized offset-assignment method for the systems that contain periodic as well as mixed messages.

6. Conclusion

The existing worst-case response-time analysis for messages with offsets in Controller Area Network (CAN) assumes that messages are queued for transmission periodically or sporadically. It does not support the analysis of mixed messages that are simultaneously time and event triggered. A mixed message can be queued both periodically and sporadically, i.e., it may not have a periodic activation pattern. Mixed messages are implemented by several high-level protocols for CAN that are used in the automotive industry today. We identified three different implementation methods for mixed messages in high-level protocols. For some implementations of mixed messages, the existing offset-based analysis still provides safe upper bounds on the response times. Whereas for the others, the existing analysis is not applicable.

We extended the existing offset-based analysis for CAN to provide safe upper bounds on the response times of mixed messages. The extended analysis is generally applicable to any high-level protocol for CAN that supports periodic, sporadic, and mixed transmission of messages. We also implemented the extended analysis as a standalone simulator that will be integrated as a plug-in with the existing industrial tool suite (Rubus-ICE). We performed a number of experiments to compare the analysis of mixed messages with and without offsets. The results indicate that it is possible to achieve up to 4.48% improvement in schedulability when mixed messages are scheduled with offsets.

In the future, we plan to conduct an industrial case study by modeling an automotive embedded application with the Rubus-ICE and analyzing it with the extended analysis plug-in. We also plan to develop an optimized offset assignment method for the systems that contain periodic as well as mixed messages.

Acknowledgement

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

References

[1] Arcticus Systems. Web page, http://www.arcticus-systems.com.

[2] Automotive networks. CAN in Automation (CiA). http://www.can-cia.org/index.php?id=416.

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

[4] CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Version 4.02. February 13, 2002. http://www.can-cia.org/index.php?id=440.

[5] Requirements on Communication, Release 3.0, Rev 7, Ver. 2.2.0. The AUTOSAR Consortium, Sep., 2010. www.autosar.org/download/R3.0/AUTOSAR_SRS_COM.pdf.

[6] Volcano Network Architect (VNA). Mentor Graphics. http://www.mentor.com/products/vnd/communication-management/vna.

[7] Hägglunds Controller Area Network (HCAN), Network Implementation Specification. *BAE Systems Hägglunds, Sweden (internal document)*, April 2009.

[8] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings. Fixed priority pre-emptive scheduling:an historic perspective. *Real-Time Systems*, 8(2/3):173–198, 1995.

[9] C. Braun, L. Havet, and N. Navet. Netcarbench: A benchmark for techniques and tools used in the design of automotive communication systems. In *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems*, Nov. 2007.

[10] 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)*, pages 181–192, Sep. 2011.

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

[12] 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.

[13] 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)*, pages 41 –45, nov. 2009.

[14] M. Grenier, L. Havet, and N. Navet. Pushing the limits of can- scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software (ERTS)*, 2008.

[15] M. Grenier, L. Havet, and N. Navet. Automotive embedded systems handbook. In N. Navet and F. Siminot-Lion, editors, *Chapter 14: Scheduling messages with offsets on Controller Area Network - a major performance boost*. CRC Press, 2009.

[16] 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*, 152(2):148–166, March 2005.

[17] ISO 11898-1. Road Vehicles interchange of digital information controller area network (CAN) for high-speed communication, ISO Standard-11898, Nov. 1993.

[18] C. Liu and J. Layland. Scheduling algorithms for multi-programming in a hard-real-time environment. *ACM*, 20(1):46–61, 1973.

[19] 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)*, pages 1–4, sept. 2011.

[20] 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.

[21] 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)*, pages 210 –221, April 2012.

[22] 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)*, pages 229–238, Sep. 2011.

[23] M. Nolin, J. Mäki-Turja, and K. Hänninen. Achieving Industrial Strength Timing Predictions of Embedded System Behavior. In *ESA*, pages 173–178, 2008.

[24] Robert Bosch GmbH. CAN Specification Version 2.0. Postfach 30 02 40, D-70442 Stuttgart, 1991.

[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] 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*, 28(2/3):101–155, 2004.

[27] A. Szakaly. Response Time Analysis with Offsets for CAN. Master's thesis, Department of Computer Engineering, Chalmers University of Technology, Nov. 2003.

[28] K. Tindell and A. Burns. Guaranteeing Message Latencies on Controller Area Network (CAN). In *1st International CAN Conference, 1994*, pages 1 –11.

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

[30] 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 (WFCS)*, May 2012.