Response-Time Analysis of Mixed Messages in Controller Area Network with Priority- and FIFO-Queued Nodes

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 Controller Area Network (CAN) is a widely used real-time network in automotive domain. We identify that the existing response-time analysis for messages in CAN with some of the connected nodes implementing priority queues while others implementing FIFO queues does not support the analysis of mixed messages. The existing analysis assumes that a message is queued for transmission either periodically or sporadically. However, a message can also be queued both periodically and sporadically using a mixed transmission mode implemented by several high-level protocols for CAN used in the industry today. We extend the existing analysis which is generally applicable to any high-level protocol for CAN (with priority- and FIFO-queued nodes) that uses periodic, sporadic, and mixed transmission of messages.

1. Introduction

Controller Area Network (CAN) [15] 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 [16]. According to CAN in Automation (CiA) [1], the estimated number of CAN enabled controllers sold in 2011 are about 850 million. In total, more than two billion CAN controllers have been sold until today. Out of this huge number, approximately 80% CAN controllers have been used in automotive applications [1]. These facts indicate the popularity of CAN in the automotive domain. It 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) [8], CANopen [3], Hägglunds Controller Area Network (HCAN) [10], CAN for Military Land Systems domain (MilCAN) [4].

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) [11, 24] 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 [22].

1.1. Related Work

The schedulability analysis of CAN was developed by Tindell et al. [27] 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 [7, 12, 20]. Furthermore, this analysis has served as the basis for many research projects. Later on, Davis et al. [13] refuted, revisited and revised the analysis developed by Tindell et al.

The communication model used in [27, 13] supports CAN messages that are queued for transmission periodically or sporadically. The analysis in [27, 13] does not support the response-times computation of mixed messages in CAN, i.e., the messages that are simultaneously time (periodic) and event triggered. In [19], Mubeen et al. extended the existing analysis for mixed messages in CAN. The extended analysis supports the worst-case response-time computation of CAN messages that are queued for transmission periodically, sporadically and both periodically and sporadically (mixed).

The analysis in [27, 13, 19] assumes that all CAN device drivers implement priority queues. This means that the highest priority message at each node enters into arbitration on the network. In [14], Davis et al. pointed out that this assumption may become invalid when some nodes in a CAN network implement FIFO queues while others implement priority queues. Hence, they extended the analysis in

[27, 13] which is applicable to the CAN network where some nodes implement priority queues and some implement FIFO queues. However, the extended analysis in [14] does not support mixed messages. In [18], we presented the basic idea and initial work regarding the extension of existing analysis of CAN with FIFO queues [14] to support mixed messages. However, it was a work in progress and lacked the detailed analysis and intuition for the extensions.

1.2. Paper Contribution

We identified that the existing RTA of CAN with FIFO queues [14] does not support the analysis of common message transmission patterns (mixed messages) which are implemented by some high-level protocols used in the industry. Moreover, the analysis of CAN for mixed messages [19] does not support FIFO-queued nodes in the systems. We extend the existing analysis of CAN with FIFO queues [14] by integrating it with the analysis of CAN for mixed messages [19].

The extended analysis is able to compute the worst-case response times of periodic, sporadic and mixed CAN messages in networks where some nodes implement priority queues while others implement FIFO queues. The existing analysis [14] places a restriction on message deadline, i.e., the deadline should be less than or equal to the period of the message. On the other hand, we assume arbitrary deadlines, i.e., the deadline of a message can be higher than its period.

The motivation for this work comes from the activity of implementing the holistic response-time analysis [26] in the existing industrial tool suite, Rubus-ICE [20, 21], that provides a component-based development environment for resource constrained distributed real-time systems while supporting several high-level protocols for CAN.

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. In Section 3, we describe the scheduling model. In Section 4, we extend the existing analysis. Section 5 concludes the paper.

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 [25]. 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 classical scheduling analysis of CAN [27, 13] which is recently extended for FIFO queues [14] 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 an event then the transmission type of a message is called mixed transmission. In other words, a mixed message is simultaneously time (periodic) and event triggered (sporadic). In [19], we identified two types of implementations of mixed messages used in the industry. In this section we revisit these methods and also discuss a third implementation method.

Consistent Terminology

To stay consistent throughout the paper, 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 [3] and "Minimum Delay Time (MDT)" in AUTOSAR communication [6].

2.1. Method 1: Implementation of a Mixed Message by CANopen

The CANopen protocol [3] supports several transmission modes including the 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 an 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 an Event Timer. Hence, the expiry of an Event Timer is considered as an additional event for queueing of a mixed message. The Event Timer is reset every time the message is queued. It should be noted that once a mixed message is queued for transmission, any additional queueing of the same message will not take place during the Inhibit Time [3]. The transmission pattern of a mixed message in CANopen is illustrated in Figure 1. The down-pointing arrows (labeled with numbers) symbolize the queueing of messages while the upward lines (labeled with alphabets) represent arrival of the events.

In Figure 1, message 1 is queued for transmission as soon as an event A arrives (assume that the Inhibit Timer was expired). In this case, the Event Timer is reset along with the Inhibit Time. As soon as the Event Timer expires, message 2 is queued for transmission and both the Event

Timer and Inhibit Time are reset again. Similarly, message 3 is queued for transmission due to the expiry of the Event Timer. When an event B arrives, message 4 is immediately queued for transmission 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 transmitted because of the expiry of Event Timer. Hence, there exists a dependency relationship between the Inhibit Time and the Event Timer.

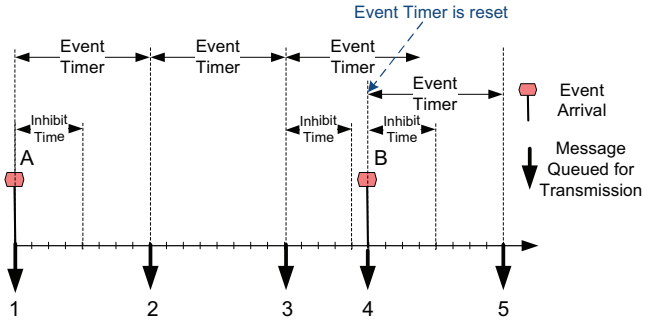

Figure 1. Mixed transmission pattern in CANopen

2.2. Method 2: Implementation of a Mixed Message by AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) [2] is a standardized software architecture for the development of automotive software. If AUTOSAR is used to develop the embedded software of a distributed real-time system that uses CAN for network communication then AUTOSAR can be viewed as a high-level protocol for CAN. The specification of AUTOSAR communication [6] is based on the OSEK communication [5] that defines several communication modes including a mixed transmission mode for network communication. Mixed transmission mode in OSEK combines the Direct (similar to the event transmission in CANopen) and the Periodic Transmission modes. Mixed transmission mode in AUTOSAR is widely used in practice. It provides an example of the second implementation method of a mixed message.

A mixed message may be queued for transmission repeatedly with a period equal to the mixed transmission mode time period. The mixed message can also be queued for transmission 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 MDT starts with every transmission of a mixed message. If further events arrive while MDT is running, their transmissions are delayed until MDT expires. Similarly, if MDT is running then the periodic transmission of a mixed message will also be delayed until MDT expires. The transmission pattern of a mixed message implemented by AUTOSAR is illustrated in Figure 2.

In Figure 2, message 1 is queued for transmission because of partly periodic nature of a mixed message. As soon as message 1 is queued, MDT is started. When an event A arrives, the message 2 is queued for transmission 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.

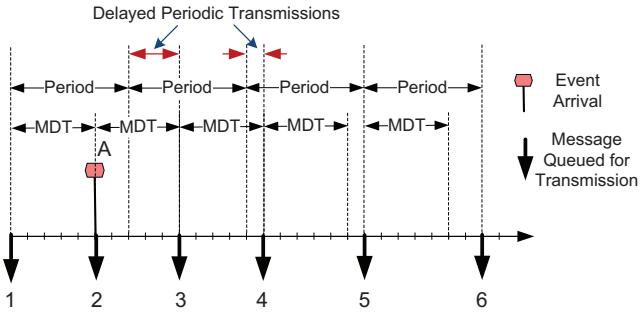

Figure 2. Mixed transmission pattern in AUTOSAR

2.3. Method 3: Implementation of a Mixed Message by HCAN

A mixed message defined by HCAN protocol [10] contains signals out of which some are periodic and some are of event type. 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 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.

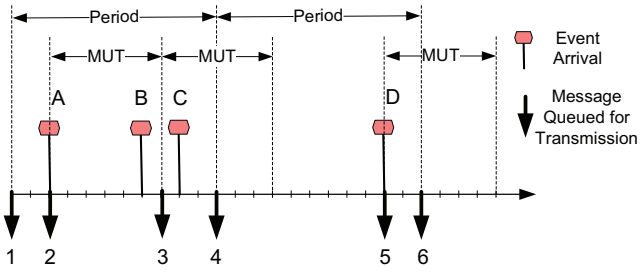

Figure 3. Mixed transmission pattern in HCAN

In Figure 3, message 1 is queued for transmission because of partly periodic nature of a mixed message. 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. It should be noted that although, MUT was not yet expired, the event signal corresponding to event C was packed in message 4 and queued as part of the periodic message. Hence, there is no need to queue an additional sporadic

message when MUT expires. This indicates that the periodic transmission of a mixed message cannot be interfered by the sporadic transmission. When an 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 and Motivation

In the first method, the Event Timer is reset every time a mixed message is queued for transmission. The most natural interpretation of a mixed message from the specification of CANopen is that there is an implicit requirement that the worst-case periodicity of transmission of a mixed message can never be higher than the Inhibit Time [3, 9, 23]. Hence, in the worst case, a mixed message is queued for transmission every time the Inhibit Timer expires.

The implementation of a mixed message in method 2 is similar to method 1 to some extent. A main difference is that in method 2, the periodic transmission can be delayed until the expiry of MDT as shown by the transmission of message 3 in Figure 2. Whereas in method 1, the periodic transmission is not delayed, in fact, the Event Timer is restarted with every sporadic transmission as shown by the transmission of messages 4 and 5 in Figure 1. The MDT timer is started with every periodic or sporadic transmission of a mixed message. Hence, it can be concluded that the worst-case periodicity of a mixed msg in method 2 can never be higher than MDT. Therefore, the original CAN analysis [27, 13] can be used for analyzing mixed messages in the first and second implementation methods. Moreover, the schedulability analysis of CAN with FIFO queues [14] is also applicable for mixed messages in CAN that are implemented using first two methods.

However, third method of implementing a mixed message is more complex because the periodic transmission is independent of the sporadic transmission. The periodic timer is not reset with every sporadic transmission. A mixed message can be queued for transmission even if MUT is not expired as shown by the transmission of message 4 in Figure 3. Hence, the worst-case periodicity of a mixed msg is neither bounded by period nor by MUT.

We extended the existing analysis [27, 13] by treating a mixed message as two separate message streams with same IDs and priorities [19]. However, the extended analysis in [19] does not support the analysis of CAN messages in a system in which some nodes implement FIFO queues. In [14], Davis et.al extended the existing analysis of CAN [27, 13] for FIFO queues. But the computed worst-case response times of mixed messages (implementation method 3) in CAN with FIFO queues will be optimistic if the existing analysis [14] is used. This calls for the need to extend the analysis of CAN with FIFO queues to support the analysis of mixed messages.

3. System Scheduling Model

The system scheduling model is an extension of the communication model that was originally developed by Tindell et al. [27] and later extended by many researches for various purposes. It basically combines the communication model of RTA of CAN with FIFO queues [14] with the communication model of RTA of CAN for mixed messages [19]. The system consists of a number of nodes connected to a single CAN network. The message queue in each node may have priority-based or FIFO-based implementation. If a node implements a priority queue, it is designated as PQ-node. On the other hand, if a node implements a FIFO queue then it is identified as FQ-node. For a PQ-node, the highest priority message in a node enters into the bus arbitration whereas, the oldest message in a node enters into the bus arbitration in a FQ-node [14].

Each CAN message m has an ID_m which is a unique identifier. 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 a standard CAN frame uses an 11-bit identifier whereas an extended CAN frame uses a 29-bit identifier. There is a $TRANSMISSION\_TYPE$ of each message that specifies whether the message is periodic or sporadic or mixed. Each message has a unique priority (P_m), transmission time (C_m) and queueing jitter (J_m) which is inherited from the task that queues the message, i.e., the sending task. J_m is equal to the difference between the worst-case and best-case response times of 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, each frame has a period, denoted by T_m. Whereas in the case of sporadic transmission, each frame has a MUT_m that refers to the minimum time that should elapse between the transmission of any two sporadic frames. Each message has a blocking time B_m which refers to the largest amount of time this message can be blocked by any lower priority message. Each message has a worst-case response time, denoted by R_m, and defined as the longest time between the queueing of the message (on the sending node) and the delivery of the message to the destination buffer (on the destination node).

A system is considered schedulable if all of its messages are schedulable. A message m is deemed schedulable if its R_m is less than or equal to its deadline D_m. We assume that the deadlines are arbitrary which means that they can be greater than the periods or minimum update times of the corresponding messages. We further assume that CAN controllers are capable of buffering more than one instance of a message.

When a message has a mixed transmission type, we duplicate the message in the analysis model. Hence, each mixed message has two copies which are treated as two separate messages. One copy is periodic while the other is sporadic. All the attributes of these duplicates including ID, priority, release jitter, transmission time and blocking time are the same except that the periodic copy inherits T_m while the sporadic copy inherits MUT_m.

If an FQ-node transmits a message m then the set of all the messages transmitted by this node is defined by $M(m)$. The Lowest priority message in $M(m)$ is defined by L_m. The sum of the transmission times of all the messages in $M(m)$ is defined by C_m^{SUM}. The transmission time of the shortest message in $M(m)$ is designated by C_m^{MIN}. f_m denotes the maximum buffering time between the instant a message m enters the FIFO queue and the instant it becomes the oldest message in the queue. It is equal to zero for a message belonging to a node that implements a priority queue [14].

4. Extended Analysis

We extend the existing analysis of CAN with both PQ-nodes and FQ-nodes [14] by adapting the schedulability analysis of CAN for mixed messages [19]. Let the message under analysis be denoted by m. We treat a message differently based on its transmission type. In order to keep the notations simple and consistent, we define a function $\xi(m)$ that represents the transmission type of a message m. Formally, the domain of this function can be defined as:

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

Where P, S and M stand for periodic, sporadic and mixed. We consider two different cases. In the first, the message under analysis is assumed to be periodic or sporadic. While in the second, the message under analysis is considered mixed.

4.1. Case: When m is a Periodic or a Sporadic Message

The worst-case response time of a message under analysis is computed differently for PQ-and FQ-nodes [14]. Once again, we consider two cases: the first case assumes that m belongs to a node that implements a priority queue; the second case considers that m belongs to a node that implements a FIFO queue.

4.1.1 Priority-Queued Messages

The worst-case response time of each instance q of a periodic or sporadic message m that is queued at a PQ-node is computed by the following equation.

$$R_m(q) = \begin{cases} J_m + \omega_m(q) - qT_m + C_m, & \text{if } \xi(k) = P \\ J_m + \omega_m(q) - qMUT_m + C_m, & \text{if } \xi(k) = S \end{cases} \quad (1)$$

If the transmission type of m is periodic then the message period is taken into account. However, if the transmission type of m is sporadic, minimum update time is used in the above equation. C_m is computed according to [13] as follows.

$$C_m = \left(g + 8s_m + 13 + \left\lfloor \frac{g + 8s_m - 1}{4} \right\rfloor \right) \tau_{bit} \quad (2)$$

In the above equation, τ_{bit} denotes the time required to transmit a single bit on CAN bus. Its value depends upon the speed of the bus. g is equal to 34 or 54 for standard or extended CAN frame formats respectively. For a Standard CAN identifier, (2) can be simplified as follows.

$$C_m = (55 + 10s_m)\tau_{bit} \quad (3)$$

Similarly, the transmission time of m for an Extended CAN identifier is given by the following equation.

$$C_m = (80 + 10s_m)\tau_{bit} \quad (4)$$

Worst-Case Queueing Delay of a Periodic or Sporadic Message

In (1), ω_m represents the worst-case queueing delay and is equal to the longest time that elapses between the instant m is queued by the sending task in the priority-ordered send queue and the instant when m starts its transmission. In other words, ω_m is the interference caused by other messages to m. The algorithms for the computation of ω_m should include the interference caused by all the other periodic, sporadic and mixed messages. The existing analysis [14] has a limitation that it considers the interference caused by only periodic and sporadic messages.

It is important to mention that CAN uses fixed-priority non-preemptive scheduling and therefore, 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. For a message queued at a PQ-node, ω_m is computed by the following fixed-point iteration.

$$\omega_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} I_k C_k \quad (5)$$

In the above equation, $hp(m)$ refers to the set of all messages in the system that have higher priority than m. The last term in (5) represents the interference from the higher priority messages. In order to solve this iterative equation, initial value of ω_m^n can be taken according to [13] as given by the following equation.

$$\omega_m^0(q) = B_m + qC_m \quad (6)$$

The iterations in (5) stop either when the queueing delays in the previous and current iterations are equal or when the response time exceeds the deadline. Since, CAN uses fixed priority non-preemptive scheduling, any message can be blocked by only one message in the set of lower priority messages. Hence, a message under analysis can only be blocked by either the periodic copy or the sporadic copy of any lower priority mixed message. It should be noted that both the copies of a mixed message have the same transmission time, C_m. Hence, B_m is equal to the largest transmission time among all periodic, sporadic and mixed messages in a set of lower priority messages with respect to m and is given by the following equation:

$$B_m = \max_{\forall k \in lp(m)} (C_k) \quad (7)$$

where, $lp(m)$ refers to the set of all messages in the system that have lower priority than m.

A higher priority message k contributes an extra delay, equal to f_k, to the worst-case queueing delay of m if k belongs to the FQ-node. f_k represents the delay after which the higher priority message k belonging to the FQ-node becomes the oldest message in the queue and can take part in the priority-based arbitration [14]. The existing analysis for mixed messages in CAN [19] does not take this additional delay into account. f_k is zero if k belongs to a PQ-node. We recommend to use the iterative algorithm to compute f_k that is proposed by Davis et al. in [14].

In (5), I_k is computed differently for different values of $\xi(k)$ (k is the index of any higher priority message) as shown below. The interference by a higher priority mixed message contains the contribution from both the duplicates.

$$I_k = \begin{cases} \left\lceil \frac{\omega_m^n(q)+J_k+f_k+\tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{P} \\ \left\lceil \frac{\omega_m^n(q)+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{S} \\ \left\lceil \frac{\omega_m^n(q)+J_k+f_k+\tau_{bit}}{T_k} \right\rceil + & \\ \left\lceil \frac{\omega_m^n(q)+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{M} \end{cases} \tag{8}$$

Length of the Busy Period

A busy period for a priority level-m is defined as the contiguous interval of time during which at least one message of priority m has not completed its transmission [13, 14]. Since, we assume arbitrary deadlines, i.e, the deadline of a message may be greater than its period, there can be multiple instances of m in the busy period. In order to calculate the worst-case response time of m, the number of instances of m that become ready for transmission before the end of the busy period should be known first. Then the response time of each instance of m should be computed. The largest value among the response times of all instances of m should be picked up as its worst-case response time.

There can be another reason to check if more than one instance of m occur in the priority level-m busy period. Since, the message transmission in CAN is non-preemptive, the transmission of previous instance of m could delay the current instance of a higher priority message that may add to the interference received by the current instance of m. This phenomenon was identified by Davis et al. and termed as "push through interference" [13]. Because of this interference, a higher priority message may be waiting for its transmission before the transmission of the current instance of m finishes. Hence, the length of busy period may extend beyond T_m or MUT_m.

The length of priority level-m busy period, denoted by t_m, is given by the following equation. The effect of extra delay from the messages belonging to the FQ-nodes is also taken into account. t_m can be computed by the following recursive equation.

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} I'_k C_k \tag{9}$$

Where, $hep(m)$ refers to the set of all messages in the system that have higher or equal priority than m. I'_k is given by the following relation. Note that the contribution of both the duplicates of a mixed message k in a set $hep(m)$ is taken into account.

$$I'_k = \begin{cases} \left\lceil \frac{t_m^n+J_k+f_k}{T_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{P} \\ \left\lceil \frac{t_m^n+J_k+f_k}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{S} \\ \left\lceil \frac{t_m^n+J_k+f_k}{T_k} \right\rceil + \left\lceil \frac{t_m^n+J_k+f_k}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{M} \end{cases} \tag{10}$$

In order to solve the recursive equation (9), C_m can be used as an initial value of t_m^n as shown below.

$$t_m^0 = C_m \tag{11}$$

The right hand side of (9) is a monotonic non-decreasing function of t_m. Equation (9) is guaranteed to converge if the bus utilization for messages of priority level-m and higher, denoted by U_m, is less than 1. That is,

$$U_m < 1 \tag{12}$$

where U_m is computed by the following equation:

$$U_m = \sum_{\forall k \in hep(m)} C_k I''_k \tag{13}$$

where I''_k is given by the following relation:

$$I''_k = \begin{cases} \frac{1}{T_k}, & \text{if } \xi(\text{k}) = \text{P} \\ \frac{1}{MUT_k}, & \text{if } \xi(\text{k}) = \text{S} \\ \frac{1}{T_k} + \frac{1}{MUT_k}, & \text{if } \xi(\text{k}) = \text{M} \end{cases} \tag{14}$$

In the above equation, the contribution by both the copies of all mixed messages belonging to the set $hep(m)$ is taken into account while calculating the bus utilization.

The number of instances of m, denoted by Q_m, that becomes ready for transmission before the busy period ends is given by the following equation (similar to the existing analysis of mixed messages):

$$Q_m = \begin{cases} \left\lceil \frac{t_m+J_m}{T_m} \right\rceil, & \text{if } \xi(\text{m}) = \text{P} \\ \left\lceil \frac{t_m+J_m}{MUT_m} \right\rceil, & \text{if } \xi(\text{m}) = \text{S} \end{cases} \tag{15}$$

The index of each message instance is identified by q. The range of q is shown as follows.

$$0 \leq q \leq Q_m - 1 \tag{16}$$

After computing the response times of all instances of m, we select the largest value among these response times to be the worst-case response time of m as shown below.

$$R_m = max(R_m(q)), \qquad \forall\, 0 \leq q \leq (Q_m - 1) \tag{17}$$

4.1.2 FIFO-Queued Messages

The existing analysis by Davis et al. [14] is FIFO-symmetric which means that all the messages belonging to FQ-node will have same upper bound for their response time. In order to derive the worst-case response time of a periodic or sporadic message belonging to FQ-node, we will consider the worst-case conditions. Hence, we will assume that the message under analysis is the lowest priority message, i.e., L_m in the group $M(m)$ with the smallest transmission time C_m^{MIN} (to maximize the interference from the messages in $M(m)$ as well as from the messages belonging to other nodes). The worst-case response time of a periodic or sporadic message m that is queued at the FQ-node is given by the following equation.

$$R_m = \begin{cases} J_m + \omega_m(q) - qT_m + C_m^{MIN}, \\ \qquad\qquad \text{if } \xi(\text{k}) = \text{P} \\ J_m + \omega_m(q) - qMUT_m + C_m^{MIN}, \\ \qquad\qquad \text{if } \xi(\text{k}) = \text{S} \end{cases} \tag{18}$$

In [14], message deadlines are assumed to be equal to or less than the corresponding periods. Hence, for an arbitrary message m belonging to $M(m)$ in the FQ-node, there could be only one instance of every other message queued ahead of m. In [14], the maximum amount of interference received by m before it becomes the oldest message in the FIFO queue and ready to take part in the priority-based arbitration is bounded by $(C_m^{SUM} - C_m^{MIN})$. This interference bound may not be applicable in our case because we assume that the messages have arbitrary deadlines which means that they can be greater than the periods or minimum update times of the corresponding messages. Therefore, it is possible to have more than one instance of any higher priority message queued ahead of m in the FIFO queue.

Interference on m from Messages in $M(m)$

Now we derive an upper bound for the number of instances of each message in the group $M(m)$ that can be queued ahead of m. Consider a simple but intuitive example as shown in Figure 4. Let the message under analysis be m (lowest priority message in $M(m)$). Also consider an arbitrary message i in the group $M(m)$. Assume both i and m are periodic and have same transmission times. We consider four different cases with respect to the relationship between message periods as shown in Figure 4. In case (a), T_i is smaller than T_m. In case (b), T_i is equal to T_m. In case (c), T_i is greater than T_m. In case (d), T_i is smaller than T_m and at the same time T_m is an integer multiple of T_i. These cases essentially cover all the cases required to derive the upper bound on the maximum number of instances of i queued ahead of any instance of m.

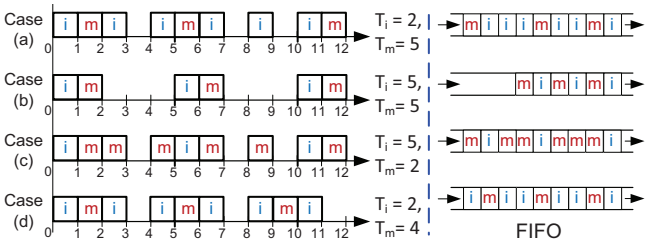

Figure 4. Demonstration of maximum interference on m from the messages in group $M(m)$

The periods of i and m in each case are shown in Figure 4. The left hand side of Figure 4 shows the time line during which each instance of i and m is queued in the FIFO queue. Whereas, the right hand side of Figure 4 depicts the corresponding FIFO queue as if none of the messages was transmitted. The maximum number of instances of i that are queued ahead of any instance of m in the FIFOs are $3, 1, 1$ and 2 in the case (a), (b), (c) and (d) respectively. Let Q_i denotes the maximum number of instances of i in the group $M(m)$ that can be queued ahead of any instance of m in the FIFO queue. We can generalize Q_i for all the cases as follows.

$$Q_i = \left\lceil \frac{T_m}{T_i} \right\rceil \tag{19}$$

Now we consider the effect of jitter of i, denoted by J_i, on the interference of m. Because of J_i, additional instances of i can be queued ahead of m. Thus, taking the effect of jitter into account, (19) can be written as:

$$Q_i = \left\lceil \frac{T_m + J_i}{T_i} \right\rceil \tag{20}$$

Since, i can be periodic, sporadic or mixed, we can generalize (20) as follows.

$$Q_i = \begin{cases} \left\lceil \frac{T_m + J_i}{T_i} \right\rceil, & \text{if } \xi(\text{i}) = \text{P} \\ \left\lceil \frac{T_m + J_i}{MUT_i} \right\rceil, & \text{if } \xi(\text{i}) = \text{S} \\ \left\lceil \frac{T_m + J_i}{T_i} \right\rceil + \left\lceil \frac{T_m + J_i}{MUT_i} \right\rceil, & \text{if } \xi(\text{i}) = \text{M} \end{cases} \tag{21}$$

Worst-Case Queueing Delay of a Mixed Message

The worst-case queueing delay, ω_m, in (18) can be computed in a similar fashion as in (5) with the addition of extra delay as shown in (21).

$$\omega_m^{n+1}(q) = B_{L_m} + \sum_{\forall i \in M(m) \wedge i \neq m} Q_i C_i + qC_m^{MIN} + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} I_k C_k \tag{22}$$

Where k is any message that has priority higher than the lowest priority message in the FQ-node in which m is

queued. Moreover, k does not belong to the FQ-node in which m is queued. i is any message, other than m, in the group $M(m)$. B_{L_m} is the blocking time of L_m which refers to the maximum transmission time of a message in a set of messages with lower priority then L_m that are sent by the other nodes. Since, the interference contributed to m by higher priority messages from other nodes (both PQ and FQ) is independent of m belonging to a PQ-node or FQ-node, I_k can be computed using (8). The initial value of ω_m^n to solve the recursive equation (22) can be selected as follows.

$$\omega_m^0 = B_{L_m} + \sum_{\forall i \in M(m) \wedge i \neq m} Q_i C_i + q C_m^{MIN} \quad (23)$$

Length of the Busy Period

The length of priority level-m busy period, denoted by t_m, can be computed in a similar fashion as in (9) and by following the intuition from (22). The effect of extra delay from the messages belonging to the FQ-nodes is also taken into account. t_m can be computed by the following recursive equation.

$$t_m^{n+1} = B_{L_m} + \sum_{\forall i \in M(m) \wedge i \neq m} Q_i C_i + \sum_{\forall k \in hep(L_m) \wedge k \notin M(m)} I'_k C_k \quad (24)$$

The initial value for t_m^n can be selected using (11). Since, the interference caused to m by higher priority messages from other nodes (both PQ and FQ) is independent of m belonging to a PQ-node or FQ-node, I'_k can be computed using (10). Similarly, the total number of instances of m that becomes ready for transmission before the busy period ends can be calculated using (15). The worst-case response time of m is the largest value of response time among all its instances as shown in (17).

4.2. Case: When m is a Mixed Message

When a message under analysis is mixed, we treat the message as two separate message streams, i.e., the mixed message is duplicated as the periodic and sporadic messages. The response time of both the duplicates is computed separately. For simplicity, we denote the periodic and sporadic copies of a mixed message m by m_P and m_E respectively. Let the worst-case response time of m_P and m_E be denoted by R_{m_P} and R_{m_E} respectively. The worst-case response time of m is equal to the largest value between R_{m_P} and R_{m_E} as given by the following equation:

$$R_m = max(R_{m_P}, \ R_{m_E}) \quad (25)$$

4.2.1 Priority-Queued Messages

For a priority-queued mixed message, the response times of each instance of m_P and m_E are computed separately by adapting the existing analysis [19]. Let us denote the total number of instances of m_P and m_E, occurring in the priority level-m busy period, by Q_{m_P} and Q_{m_E} respectively. Assume that the index variable for message instances of m_P and m_E is denoted by q_{m_P} and q_{m_E} respectively. The range of q_{m_P} and q_{m_E} is shown by the following equations:

$$0 \leq q_{m_P} \leq (Q_{m_P} - 1) \quad (26)$$

similarly,

$$0 \leq q_{m_E} \leq (Q_{m_E} - 1) \quad (27)$$

The worst-case response time of m_P is equal to the largest value among the response times of all of its instances occurring in the busy period as shown by the following equation.

$$R_{m_P} = max(R_{m_P}(q_{m_P})) \quad (28)$$

Similarly, the worst-case response time of m_E is equal to the largest value among the response times of all of its instances occurring in the busy period. It is given by the following equation.

$$R_{m_E} = max(R_{m_E}(q_{m_E})) \quad (29)$$

The worst-case response time of each instance of m_P and m_E can be derived by adapting the equations for the computation of worst-case response time of periodic and sporadic messages respectively (derived in the first case) as given by the following two equations:

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P} T_m + C_m \quad (30)$$

$$R_{m_E}(q_{m_E}) = J_m + \omega_{m_E}(q_{m_E}) - q_{m_E} MUT_m + C_m \quad (31)$$

The queueing jitter, J_m, is the same (equal) in both the equations (30) and (31). The transmission time, C_m, is also the same in these equations and is calculated using (3) or (4) depending upon the type of CAN frame identifier. Although, both the duplicates of m inherit same J_m and C_m from it, they experience different amount of worst-case queueing delay caused by other messages.

Worst-Case Queueing Delay of a Mixed Message

The worst-case queueing delay experienced by m_P and m_E is denoted by ω_{m_P} and ω_{m_E} in (30) and (31) respectively. ω_{m_P} and ω_{m_E} can be computed by adapting the equation for the computation of worst-case queueing delay in (5). However, in this equation we need to add 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_P and m_E have equal priorities, any number of instances of m_P queued ahead of m_E will contribute an extra delay to the worst-case queueing delay experienced by m_E and vice versa. We will reuse the effect of self interference in a mixed message that we derived in [19]. The worst-case queueing delay for m_P and m_E can be computed using the following equations.

$$\omega_{m_P}^{n+1}(q_{m_P}) = B_m + q_{m_P} C_m + \sum_{\forall k \in hp(m)} I_{k_P} C_k + Q_{m_E}^P C_m \quad (32)$$

$$\omega_{m_E}^{n+1}(q_{m_E}) = B_m + q_{m_E}C_m + \sum_{\forall k \in hp(m)} I_{k_E}C_k + Q_{m_P}^{E}C_m \quad (33)$$

The effect of self interference can be seen in the last terms of (32) and (33). $Q_{m_E}^{P}$ denotes the total number of instances of m_E that are queued ahead of $q_{m_P}^{th}$ instance of m_P. Similarly, $Q_{m_P}^{E}$ denotes the total number of instances of m_P that are queued ahead of $q_{m_E}^{th}$ instance of m_E. The values of $Q_{m_E}^{P}$ and $Q_{m_P}^{E}$ are computed as follows.

$$Q_{m_E}^{P} = \left\lceil \frac{q_{m_P}T_m + J_m}{MUT_m} \right\rceil \quad (34)$$

$$Q_{m_P}^{E} = \left\lceil \frac{q_{m_E}MUT_m + J_m}{T_m} \right\rceil \quad (35)$$

In order to solve the recursive equations (32) and (33), the initial values of $\omega_{m_P}^{n}(q_{m_P})$ and $\omega_{m_E}^{n}(q_{m_E})$ can be selected according to (6) in a similar fashion. I_{k_P} and I_{k_E} are computed according to the following fixed-point iterations.

$$I_{k_P} = \begin{cases} \left\lceil \frac{\omega_{m_P}^{n}(q_{m_P})+J_k+f_k+\tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(k) = P \\ \left\lceil \frac{\omega_{m_P}^{n}(q_{m_P})+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = S \\ \left\lceil \frac{\omega_{m_P}^{n}(q_{m_P})+J_k+f_k+\tau_{bit}}{T_k} \right\rceil + & \\ \left\lceil \frac{\omega_{m_P}^{n}(q_{m_P})+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = M \end{cases} \quad (36)$$

$$I_{k_E} = \begin{cases} \left\lceil \frac{\omega_{m_E}^{n}(q_{m_E})+J_k+f_k+\tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(k) = P \\ \left\lceil \frac{\omega_{m_E}^{n}(q_{m_E})+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = S \\ \left\lceil \frac{\omega_{m_E}^{n}(q_{m_E})+J_k+f_k+\tau_{bit}}{T_k} \right\rceil + & \\ \left\lceil \frac{\omega_{m_E}^{n}(q_{m_E})+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = M \end{cases} \quad (37)$$

The values of I_{k_P} and I_{k_E} in (36) and (37) differ from those computed in [19] in a way that we consider an extra jitter, i.e., f_k from every message that belongs to the FQ-node.

Length of the Busy Period

The length of priority level-m busy period, denoted by t_m, can be computed by using (9) that was developed for periodic and sporadic messages in a PQ-node. This is because (9) takes into account the effect of queueing delay from all the higher and equal priority messages. Since, the duplicates of a mixed message inherit the same priority from it, the contribution of queueing delay from the duplicate is also covered in (9). Therefore, there is no need to compute t_m for m_P and m_E separately. It should be computed only once for a mixed message.

Although t_m is the same for m_P and m_E, the number of instances of both the messages that become ready for transmission just before the end of busy period, i.e., Q_{m_P} and Q_{m_E} respectively, may be different. The reason is that the computation of Q_{m_P} and Q_{m_E} require T_m and MUT_m respectively and which may have different values. Q_{m_P} and Q_{m_E} can be computed by adapting (15) that was derived for the computation of the number of instances of periodic and sporadic messages that become ready for transmission before the end of the busy period. Q_{m_P} and Q_{m_E} are given by the following equations.

$$Q_{m_P} = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \quad (38)$$

$$Q_{m_E} = \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil \quad (39)$$

4.2.2 FIFO-Queued Messages

The worst-case response time of each instance of m_P and m_E queued at the FQ-node is computed similar to the case of FIFO-queued messages that are periodic or sporadic as discussed in (18).

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P}T_m + C_m^{MIN} \quad (40)$$

$$R_{m_E}(q_{m_E}) = J_m + \omega_{m_E}(q_{m_E}) - q_{m_E}MUT_m + C_m^{MIN} \quad (41)$$

Worst-Case Queueing Delay of a Mixed Message

The algorithm for the worst-case queueing delay, ω_m, for m_P and m_E are computed by adapting the algorithms in (22), (32) and (33) as follows.

$$\omega_{m_P}^{n+1}(q_{m_P}) = B_{L_m} + \sum_{\forall i \in M(m) \wedge i \neq m} Q_i C_i + q_{m_P}C_m^{MIN} + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} I_{k_P}C_k + Q_{m_E}^{P}C_m \quad (42)$$

$$\omega_{m_E}^{n+1}(q_{m_E}) = B_{L_m} + \sum_{\forall i \in M(m) \wedge i \neq m} Q_i C_i + q_{m_E}C_m^{MIN} + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} I_{k_E}C_k + Q_{m_P}^{E}C_m \quad (43)$$

Since, the interference caused by higher priority messages from other PQ- and FQ-nodes is independent of the mixed message m belonging to a PQ-node or FQ-node, I_{k_P} and I_{k_E} can be computed using (36) and (37). The initial values of ω_{m_P} and ω_{m_E} can be computed by using (23) while considering the respective index of each instance of m_P and m_E. The value of Q_i is computed using (21) similar to the case of FIFO queued periodic or sporadic messages. The values of $Q_{m_E}^{P}$ and $Q_{m_P}^{E}$ are computed using (34) and (35) which were derived for a mixed message in a PQ-node. $Q_{m_E}^{P}$ denotes the total number of instances of m_E that are queued ahead of $q_{m_P}^{th}$ instance of m_P. Therefore, we consider only queueing jitter (34) and do not take into account any additional delay that may occur after queueing of m_P such as f_m. Similar arguments hold for $Q_{m_P}^{E}$.

Length of the Busy Period

The length of priority level-m busy period, denoted by t_m, can be computed by using (24) that was developed for periodic and sporadic messages in a FQ-node. This is because (24) takes into account the effect of queueing delay from all the higher and equal priority messages. Since, the duplicates of a mixed message inherit the same priority from it, the contribution of queueing delay from the duplicate is also covered in (24). Therefore, there is no need to compute t_m for m_P and m_E separately. It should be computed only once for a mixed message.

Although the length of the busy period is the same, the number of instances of m_P and m_E that become ready for transmission just before the end of busy period, i.e., Q_{m_P} and Q_{m_E} respectively, may be different. Q_{m_P} and Q_{m_E} can be computed by following the same reasoning and by using the equations that we derived for a mixed message in the PQ-node using (38) and (39) respectively.

5. Conclusion

The existing worst-case response-time analysis of messages in Controller Area Network (CAN) with Priority-queued and FIFO-queued nodes assumes that the 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 industry. We identified different implementation methods for mixed transmission mode in several high-level protocols. For some implementations of a mixed message, the existing analysis still provides safe upper bounds for worst-case response times. Whereas for the others, the existing analysis computes optimistic worst-case response times of mixed messages. We extended the existing analysis of CAN with FIFO queues to provide safe upper bounds on the worst-case 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 in a system comprising of Priority-queued and FIFO-queued nodes.

In the future, we plan to implement the extended analysis in the existing industrial tool suite, i.e., Rubus-ICE [17]. We also plan to conduct an industrial case study by modeling an automotive embedded application and analyzing it with the implemented analysis.

Acknowledgement

This work is supported by the Swedish Knowledge Foundation (KKS) within the projects FEMMVA and EEMDEF, the Swedish Research Council (VR) within project TiPCES, and the Strategic Research Foundation (SSF) with the centre PROGRESS. The authors would like to thank the industrial partners Arcticus Systems and BAE Systems Hägglunds, Sweden.

References

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

[2] AUTOSAR Techincal Overview, Version 2.2.2. AUTOSAR – AUTomotive Open System ARchitecture, Release 3.1, The AUTOSAR Consortium, Aug., 2008. http://autosar.org.

[3] 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.

[4] MilCAN (CAN for Military Land Systems domain). http://www.milcan.org/.

[5] OSEK/VDX Communication Version. 3.0.3, July, 2004. http://portal.osek-vdx.org/files/pdf/specs/osekcom303.pdf.

[6] 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.

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

[8] CAL, CAN Application Layer for Industrial Applications, CiA Draft Standard DS-207, Ver. 1.1. *CAN-in-Automation*, Feb. 1996.

[9] CANopen high-level protocol for CAN-bus, Version 3.0. *NIKHEF, Amsterdam*, March 2000. http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen.pdf.

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

[11] 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.

[12] L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg. Volcano - a revolution in on-board communications. In *Volvo Technology Report*, 1998.

[13] 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.

[14] 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.

[15] R. B. GmbH. CAN Specification Version 2.0. Postfach 30 02 40, D-70442 Stuttgart, 1991.

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

[17] K. Hänninen et.al. Framework for real-time analysis in Rubus-ICE. In *13th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 782 –788, 2008.

[18] 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.

[19] 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.

[20] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Support for Holistic Response-time Analysis in an Industrial Tool Suite: Implementation Issues, Experiences and a Case Study. In *19th IEEE Conference on Engineering of Computer Based Systems (ECBS)*, April 2012.

[21] 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), 2011*, pages 229 –238, Sep. 2011.

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

[23] O. Pfeiffer, A. Ayre, and C. Keydel. *Embedded Networking with CAN and CANopen*. Annabooks, 2003.

[24] 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.

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

[26] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocess. Microprogram.*, 40:117–134, April 1994.

[27] 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.