Received March 12, 2014, accepted April 14, 2014, date of publication April 22, 2014, date of current version May 6, 2014.

Digital Object Identifier 10.1109/ACCESS.2014.2319255

Extending Worst Case Response-Time Analysis for Mixed Messages in Controller Area Network With Priority and FIFO Queues

SAAD MUBEEN[1,2], (Student Member, IEEE), JUKKA MÄKI-TURJA[1,2], (Member, IEEE), and MIKAEL SJÖDIN[1], (Member, IEEE)
[1]Mälardalen University, Västerås 721 23, Sweden
[2]Arcticus Systems AB, Järfälla 175 43, Sweden

Corresponding author: S. Mubeen (saad.mubeen@mdh.se)

This work was supported in part by the Swedish Knowledge Foundation through the FEMMVA and EEMDEF Projects, in part by the Swedish Research Council through the SynthSoft and TiPCES Projects, and in part by the Strategic Research Foundation through the Centre Progress.

ABSTRACT The existing worst case response-time analysis for controller area network (CAN) with nodes implementing priority and First In First Out (FIFO) queues does not support mixed messages. It assumes that a message is queued for transmission either periodically or sporadically. However, a message can also be queued both periodically and sporadically using mixed transmission mode implemented by several higher level protocols for CAN that are used in the automotive industry. We extend the existing analysis for CAN to support any higher level protocol for CAN that uses periodic, sporadic, and mixed transmission of messages in the systems where some nodes implement priority queues, whereas others implement FIFO queues. In order to provide a proof of concept, we implement the extended analysis in a free tool, conduct an automotive-application case study, and perform comparative evaluation of the extended analysis with the existing analysis.

INDEX TERMS Controller area network, CAN protocol, real-time network, response-time analysis, distributed embedded systems, schedulability analysis, FIFO queues, mixed messages.

I. INTRODUCTION

The Controller Area Network (CAN) [1] is a widely used real-time network protocol in the automotive domain. In 2003, the International Organization for Standardization (ISO) standardized CAN in ISO 11898-1 [2]. It is a multi-master, event-triggered, serial communication bus protocol supporting bus speeds of up to 1 Mbit/s. CAN with Flexible Data-rate (CAN FD) [3] is a new protocol based on CAN that can achieve bus speed of more than 1 Mbit/s. According to CAN in Automation (CiA) [4], 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 the automotive applications. For example, there can be as many as 20 CAN networks[1] used in a modern heavy truck, while the number of CAN messages transmitted over these networks can be over 6000 [5]. These facts and figures indicate the popularity of CAN in the automotive domain. It is also used in other domains such as industrial control, medical equipments, maritime electronics, production machinery, and many others. There are a number of higher-level protocols for CAN that are developed for many industrial applications such as CAN Application Layer (CAL) [6], CANopen [7], Hägglunds Controller Area Network (HCAN) [8], and CAN for Military Land Systems domain (MilCAN) [9].

CAN finds its applications in the systems that have real-time requirements. This means that the time for response to some stimulus is as crucial as logical correctness of the response. In other words, logically correct but late response may be considered as bad as logically incorrect response. Hence, the providers of these systems are required to ensure that the actions by the systems will be taken at times that are appropriate to their environment. In order to provide evidence that each action by the system will be provided in a timely manner, *a priori* analysis techniques, such as schedulability

[1]Since, CAN uses bus topology, we use the terms network and bus interchangeably throughout the paper.

analysis [10]–[12], have been developed by the research community. Response-Time Analysis (RTA) [10]–[13] 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. 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 [14].

A. EXTENDED VERSION

This paper extends our previous work that was presented in the 9th IEEE International Workshop on Factory Communication Systems (WFCS 2012) [15]. The workshop paper presents the response-time analysis for mixed messages in CAN with FIFO queues. However, it lacks the calculations for maximum buffering time in the FIFO queues which is an important factor in the response-time calculations. Moreover, it does not evaluate and compare the extended analysis with the other related analyses. In the extended version of the paper, we generalize the analysis, by complementing it with the algorithm to calculate maximum buffering time in the FIFO queues. Moreover, we implement the extended analysis in a freely-available tool. We also show the applicability of the extended analysis by conducting an automotive-application case study. We also perform extensive evaluation of the extended analysis.

B. RELATED WORKS

Tindell et al. [16] developed the schedulability analysis for CAN. It has been implemented in the automotive industrial tools such as Volcano Network Architect (VNA) [17]. Davis et al. [18] found the analysis to be flawed in some cases. Accordingly, they revisited and revised the original analysis. The revised analysis is also implemented in the existing industrial tool suite Rubus-ICE [19], [20] which is used by several international companies.

The scheduling model used in [16] and [18] assumes that the messages are queued for transmission either periodically or sporadically. These analyses do not support the response time calculations for mixed messages in CAN, i.e., the messages that are simultaneously time (periodic) and event triggered. Mixed messages are implemented by several higher-level protocols based on CAN that are used in the automotive industry. Mubeen et al. [21] extended the seminal analysis [16], [18] to support the worst-case response-time calculations for mixed messages in CAN.

However, the analyses in [16], [18], and [21] assume that the device drivers in the CAN controllers implement priority-based queues. This means that the highest priority message at each node[2] enters into the bus arbitration. This assumption may become invalid when some controllers in the network implement FIFO queues. Some examples of the CAN controllers implementing FIFO queues are Infineon XC161CS, Microchip PIC32MX, Renesas R32C/160 and XILINX LogiCORE IP AXI Controller [22]–[24]. Davis et al. [22], [25] extended the analysis for CAN where some nodes implement priority queues while others implement FIFO queues.

In the works in [22] and [25], the message deadlines are assumed to be smaller than or equal to the corresponding periods. In [26], Davis et al. lifted this assumption by supporting the analysis for CAN messages with arbitrary deadlines. Furthermore, they extended their previous works to support RTA for CAN with FIFO and work-conserving queues. However, the analyses for CAN with FIFO queues do not support mixed messages.

C. PAPER CONTRIBUTIONS AND MOTIVATION

We identified that the existing RTA for CAN with FIFO queues [22], [25], [26] does not support the analysis of common message transmission patterns, i.e., mixed messages. These type of messages are implemented by some higher-level protocols for CAN that are used in the automotive industry. Further, the existing analysis for mixed messages in CAN [21] does not support the analysis of the systems containing nodes that implement FIFO queues. We extend the existing analysis for CAN with FIFO queues [22], [25], [26] by integrating it with the analysis for mixed messages in CAN with priority queues [21]. Moreover, we generalize the extended analysis for CAN with FIFO queues by presenting the algorithm for the calculations of maximum buffering time in the FIFO queues. The relationship between the existing and extended analyses is shown in Figure 1.

The extended analysis does not put any restrictions on the message deadlines, i.e., the deadline of a message can be lower, equal, or higher than its transmission period. The extended analysis is able to calculate 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. We also implement the extended analysis in a freely-available tool [27]. Furthermore, we show the applicability of the extended analysis by conducting the automotive-application case study. We also perform extensive evaluation of the extended analysis.

The motivation for this work comes from the industrial requirements and the activity of implementing the holistic response-time analysis [28] in the existing industrial tool suite, Rubus-ICE [20]. This tool provides a model- and component-based development environment for resource-constrained automotive distributed real-time systems while supporting several higher-level protocols based on CAN.

D. PAPER LAYOUT

The rest of the paper is organized as follows. In Section II, we discuss mixed transmission patterns supported by several higher-level protocols for CAN. In Section III, we discuss some common queueing policies in the transmit buffers of

[2]It should be noted that a node or ECU contains a CAN controller. We overload the terms node, processor, Electronic Control Unit (ECU), and CAN controller throughout the paper.

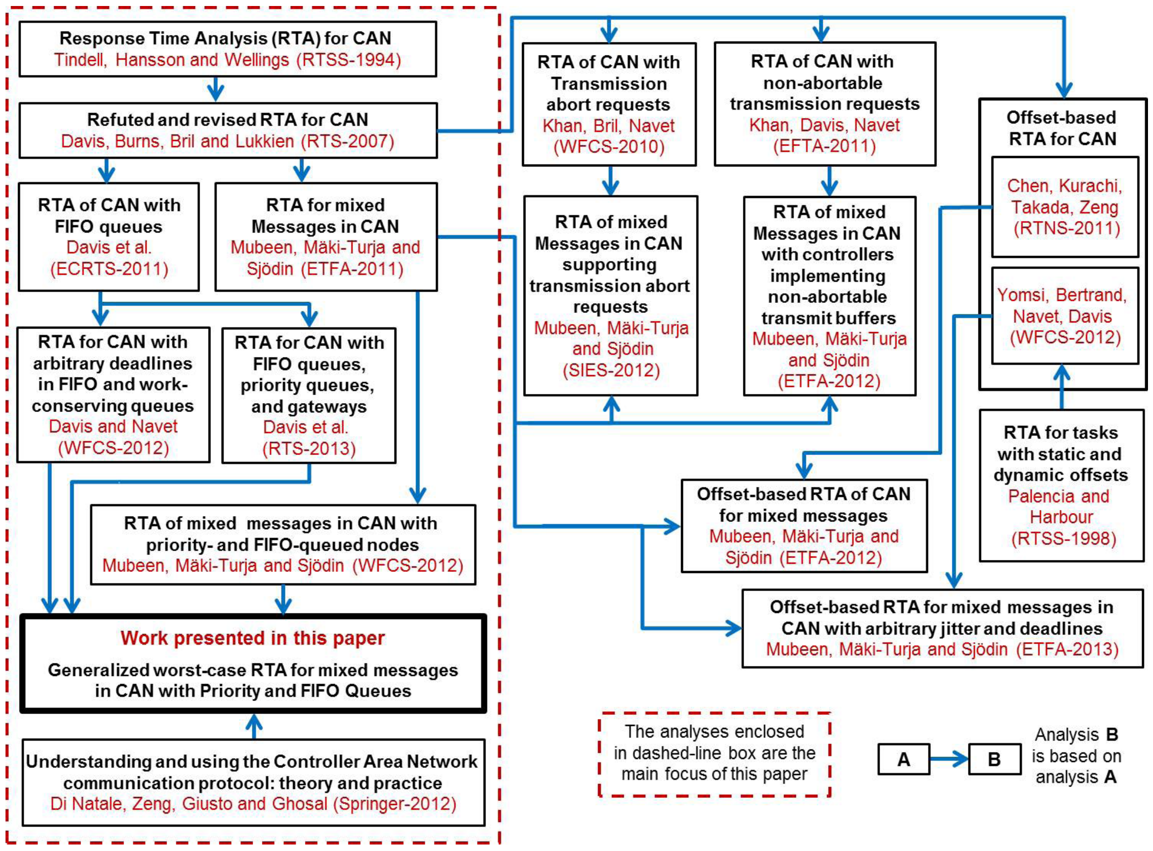
Response Time Analysis (RTA) for CAN
Tindell, Hansson and Wellings (RTSS-1994)
Refuted and revised RTA for CAN
Davis, Burns, Bril and Lukkien (RTS-2007)
RTA of CAN with FIFO queues
Davis et al. (ECRTS-2011)
RTA for mixed Messages in CAN
Mubeen, Mäki-Turja and Sjödin (ETFA-2011)
RTA for CAN with arbitrary deadlines in FIFO and work-conserving queues
Davis and Navet (WFCS-2012)
RTA for CAN with FIFO queues, priority queues, and gateways
Davis et al. (RTS-2013)
RTA of mixed messages in CAN with priority- and FIFO-queued nodes
Mubeen, Mäki-Turja and Sjödin (WFCS-2012)
Work presented in this paper
Generalized worst-case RTA for mixed messages in CAN with Priority and FIFO Queues
Understanding and using the Controller Area Network communication protocol: theory and practice
Di Natale, Zeng, Giusto and Ghosal (Springer-2012)
RTA of CAN with Transmission abort requests
Khan, Bril, Navet (WFCS-2010)
RTA of CAN with non-abortable transmission requests
Khan, Davis, Navet (EFTA-2011)
Offset-based RTA for CAN
Chen, Kurachi, Takada, Zeng (RTNS-2011)
Yomsi, Bertrand, Navet, Davis (WFCS-2012)
RTA of mixed Messages in CAN supporting transmission abort requests
Mubeen, Mäki-Turja and Sjödin (SIES-2012)
RTA of mixed Messages in CAN with controllers implementing non-abortable transmit buffers
Mubeen, Mäki-Turja and Sjödin (ETFA-2012)
RTA for tasks with static and dynamic offsets
Palencia and Harbour (RTSS-1998)
Offset-based RTA of CAN for mixed messages
Mubeen, Mäki-Turja and Sjödin (ETFA-2012)
Offset-based RTA for mixed messages in CAN with arbitrary jitter and deadlines
Mubeen, Mäki-Turja and Sjödin (ETFA-2013)
The analyses enclosed in dashed-line box are the main focus of this paper
A
B
Analysis B is based on analysis A

FIGURE 1. Relationship between the existing and extended analyses for CAN.

the CAN controllers. In Section IV, we describe the scheduling model. In Section V, we extend the existing analysis. Section VI presents the case study and evaluation of the extended analysis. Finally, Section VII summarizes and concludes the paper.

II. MIXED TRANSMISSION PATTERNS SUPPORTED BY THE HIGHER-LEVEL PROTOCOLS FOR CAN

In order to be consistent throughout the paper, we use the terms message and frame interchangeably. This is because we only consider messages that fit into one frame, i.e., the maximum size of a message can be 8 bytes. If a message is queued for transmission at periodic intervals, we use the term ''Period'' to refer to its periodicity. On the other hand, a sporadic message is queued for transmission as soon as a sporadic event occurs that changes the value of one or more signals contained in the message provided the Minimum Update Time (MUT^3) between the queueing of two successive sporadic messages has elapsed. The seminal RTA for CAN [16] and most of its extensions assume that the tasks queueing CAN messages are invoked either periodically or sporadically. However, there are some higher-level protocols and commercial extensions of CAN in which the tasks that queue the messages can be invoked periodically as well as sporadically. If a message is queued for transmission periodically as well as sporadically, the transmission type of a message is called mixed. That is, a mixed message is simultaneously time- and event-triggered. We identify three different types of implementations of the mixed messages by the higher-level protocols for CAN that are used in the automotive industry.

[3]We overload the term *MUT* to refer to the *Inhibit Time* in the CANopen protocol [7] and the *Minimum Delay Time (MDT)* in the AUTOSAR communication [29].

A. IMPLEMENTATION OF MIXED MESSAGE IN THE CANopen PROTOCOL

The CANopen protocol [7] supports mixed transmission that corresponds to the Asynchronous Transmission Mode coupled with Event Timer. The Event Timer is used for cyclic transmission of an asynchronous message. The mixed message in this protocol can be queued for transmission at the arrival of a sporadic 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. The mixed message can also be queued periodically when the Event Timer expires. The Event Timer is reset every time the message is queued. Once the mixed message is queued, any additional queueing of this message will not take place during the Inhibit Time [7]. The transmission pattern of the mixed message in the CANopen protocol is illustrated in Figure 2(a). The down-pointing arrows show queueing of

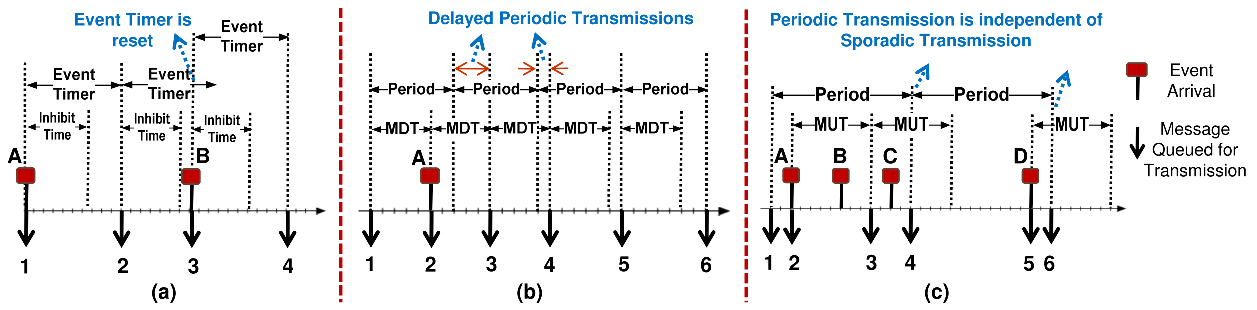

FIGURE 2. Mixed transmission pattern in higher-level protocols for CAN. (a) Mixed message in CANopen. (b) Mixed message in AUTOSAR. (c) Mixed message in HCAN.

the message while the numbers below them represent the instance number of the queued message. The upward lines labeled with alphabetic characters represent the arrival of events. Instance 1 of the mixed message 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, instance 2 is queued due to periodicity and both the Event Timer and Inhibit Time are reset again. Instance 3 of the mixed message is immediately queued upon arrival of the event *B* because the Inhibit Time has already expired. Note that the Event Timer is also reset at the same time when instance 3 is queued as shown in Figure 2(a). The instance 4 of the mixed message is queued because of the expiry of the Event Timer. There exists a dependency relationship between the Inhibit Time and the Event Timer, i.e., the Event Timer is reset with every sporadic transmission.

B. IMPLEMENTATION OF MIXED MESSAGE IN THE AUTOSAR COMMUNICATIONS

AUTOSAR (AUTomotive Open System ARchitecture) [30] can be viewed as a higher-level protocol if it uses CAN for network communication. Mixed transmission in AUTOSAR is widely used in practice. In this protocol, a mixed message can be queued for transmission periodically with the mixed transmission mode time period. The mixed message can also be queued at the arrival of an event provided the Minimum Delay Time (*MDT*) has been expired. However, each transmission of the mixed message, regardless of being periodic or sporadic, is limited by the *MDT* timer. This means that both periodic and sporadic transmissions will always be delayed until the expiry of the *MDT* timer. Figure 2(b) shows the transmission pattern of the mixed message implemented by AUTOSAR. The *MDT* timer is started as soon as the first instance of the mixed message is queued due to partly periodic nature of the mixed message. Its second instance is queued immediately upon arrival of the event *A* because the *MDT* timer has already expired. The next periodic transmission is scheduled 2 time units after the transmission of instance 2. However, the next two periodic transmissions corresponding to instances 3 and 4 are delayed because the *MDT* timer is still running. The transmissions that are delayed due to non-expiry of the *MDT* timer are identified in Figure 2(b). The periodic transmissions corresponding to instances 5 and 6 take place at the scheduled times because the *MDT* timer is already expired in both cases.

C. IMPLEMENTATION OF MIXED MESSAGE IN THE HCAN PROTOCOL

The mixed message in the HCAN protocol [8] contains signals out of which some are periodic and some are sporadic. The mixed message is queued for transmission not only periodically, but also as soon as an event occurs that changes the value of one or more event signals, provided the *MUT* between the queueing of two successive sporadic instances of the mixed message has elapsed. Hence, the transmission of the mixed message due to arrival of events is constrained by the *MUT*. The transmission pattern of mixed message in the HCAN protocol is illustrated in Figure 2(c). Instance 1 of the mixed message is queued because of periodicity. As soon as event *A* arrives, instance 2 is queued. When event *B* arrives, the next instance of the mixed message is not queued immediately because the *MUT* is not expired yet. As soon as the *MUT* expires, the third instance is queued. The third instance contains the signal changes that correspond to event *B*. Similarly, the next instance of the mixed message is not immediately queued when the event *C* arrives because the *MUT* is not expired. Instance 4 of the mixed message is queued because of periodicity. Although, the *MUT* was not expired, the event signal corresponding to event *C* was packed in instance 4 and queued as part of the periodic message. Hence, there is no need to queue an additional sporadic instance of the mixed message when the *MUT* expires. This indicates that the periodic transmission of a mixed message cannot be interfered by its sporadic transmission. This is a unique property of the HCAN protocol. When the event *D* arrives, a sporadic instance of the mixed message is immediately queued as message 5 because the *MUT* has already expired. Instance 6 is queued due to partly periodic nature of the mixed message.

D. COMPARISON OF THE THREE IMPLEMENTATIONS OF MIXED MESSAGE

In the first implementation method, the Event Timer is reset every time the mixed message is queued for transmission.

The implementation of the mixed message in method 2 is similar to method 1 to some extent. The main difference is that the periodic transmission can be delayed until the expiry of the *MDT* in method 2. Whereas in method 1, the periodic transmission is not delayed, in fact, the Event Timer is restarted with every sporadic transmission. The *MDT* timer is started with every periodic or sporadic transmission of the mixed message. Hence, the worst-case periodicity of the mixed message in methods 1 and 2 can never be higher than the Inhibit Timer and the *MDT* respectively. Therefore, the existing analyses for CAN with FIFO queues [22], [25], [26] hold intact. However, the periodic transmission is independent of the sporadic transmission in the third implementation method. The periodic timer is not reset with every sporadic transmission. The mixed message can be queued for transmission even if the *MUT* is not expired. The worst-case periodicity of the mixed message is neither bounded by the period nor by the *MUT*. Therefore, the existing analyses for CAN with FIFO queues [22], [25], [26] cannot be applied to the mixed messages in the third implementation method.

III. COMMON QUEUEING POLICIES USED IN THE CAN CONTROLLERS

The timing behavior of CAN messages is influenced by many factors including the type of queueing polices implemented by the CAN device drivers and communication stack. The most common queueing policies in the nodes connected to the CAN network are priority- and FIFO-ordered policies.

A. PRIORITY-ORDERED QUEUES

The CAN protocol implements priority-based arbitration for the transmission of messages on the network. This means, each node selects the highest priority message from its transmit buffers while entering into the bus arbitration. The highest priority message among the messages selected from each node wins the arbitration, i.e., the right to transmit over the network. Intuitively, the most natural queueing policy suited to CAN controllers is priority-ordered queueing.

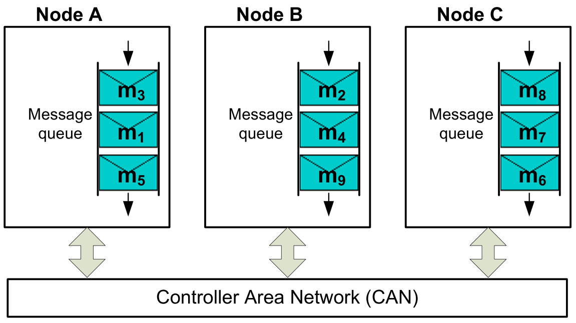

FIGURE 3. Example to demonstrate the effect of queueing policy on message transmission.

Let us consider an example to demonstrate the priority-based queueing policy as shown in Figure 3. Let there be three nodes namely Node A, Node B and Node C that are connected to a single CAN network. Each node sends three messages over the network. Node A sends the messages m_1, m_3 and m_5; Node B sends the messages m_2, m_4 and m_9; whereas, Node C sends the messages m_6, m_7 and m_8. The subscript in the name of a message represents its priority. We assume that the smaller the value of the subscript, the higher the priority of the message. Intuitively, m_1 is the highest priority message, whereas, m_9 is the lowest priority message in the system.

In order to simplify the example, assume that the transmission periods of all messages are very high compared to their transmission times. Assume that all messages in each node are queued for transmission. We also assume that there cannot be multiple instances of a message queued for transmission at the same time.

Let the nodes implement priority queues. Each node selects the highest priority message from its queue to enter into bus arbitration. In the first round, Nodes A, B, and C select messages m_1, m_2 and m_6 respectively. Message m_1 wins the arbitration and is transmitted over the network as shown in Figure 4. In the second round, Nodes A, B, and C pick messages m_3, m_2 and m_6 respectively. This time, message m_2 wins the arbitration and is transmitted over the network. Similar priority-based selection and arbitration continue during the rest of the rounds as shown in Figure 4.

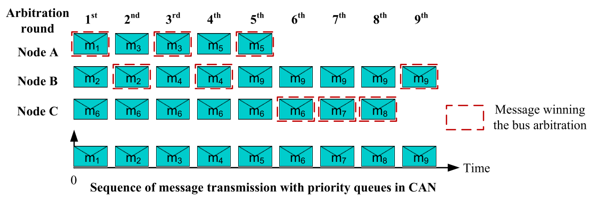
Arbitration round
1st
2nd
3rd
4th
5th
6th
7th
8th
9th
Node A
Node B
Node C
Message winning the bus arbitration
Time
0
Sequence of message transmission with priority queues in CAN

FIGURE 4. Demonstration of CAN arbitration and priority-based queueing.

B. FIFO QUEUES

The main advantages of FIFO queueing policy is that it is simple to implement and use. Some examples of the CAN controllers that implement FIFO queueing policy are Microchip PIC32MX, Infineon XC161CS, Renesas R32C/160 and XILINX LogiCORE IP AXI Controller [22], [23]. When nodes implement FIFO queues, the oldest message in the transmit queue of each node competes for the network with the oldest messages in the transmit queues in the rest of the nodes. It should be noted that even in the case of FIFO queues, the bus arbitration among CAN messages from different nodes is done on priority basis. Let us consider the three nodes, shown in Figure 3, implement FIFO queues. Intuitively, each node selects the oldest message in its queue to enter into the bus arbitration. In the first round, Nodes A, B, and C pick messages m_5, m_9 and m_6 respectively. Due to higher priority, message m_5 wins the arbitration and is transmitted over the network as shown in Figure 5. In the second round, Nodes A, B, and C pick messages m_1, m_9 and m_6 respectively. In this round, message m_1 wins the arbitration and is transmitted over the network. Similar FIFO selection and priority-based arbitration occur during the rest of the rounds as shown in Figure 5.

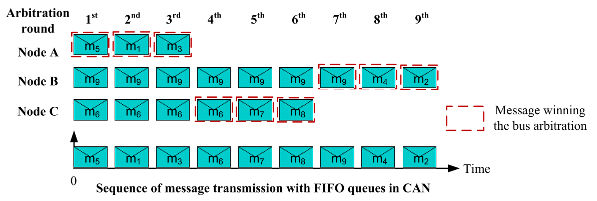
Arbitration round
1st
2nd
3rd
4th
5th
6th
7th
8th
9th
Node A
Node B
Node C
m5
m1
m3
m9
m4
m2
m6
m7
m8
Message winning the bus arbitration
Time
0
Sequence of message transmission with FIFO queues in CAN

FIGURE 5. CAN arbitration and FIFO-based queueing.

C. EFFECT OF QUEUEING POLICY ON THE RESPONSE TIMES OF MESSAGES

When FIFO queues are used, the priorities of messages are often not respected in the transmit queue within a node, e.g., the lower priority message m_5 is transmitted before the highest priority message m_1 as shown in Figure 5. As a result, priority inversion can occur due to which higher priority messages may have very large response times. This becomes evident by comparing the response time of message m_2 in the systems with priority and FIFO queues as shown in Figure 4 and Figure 5 respectively.

IV. SYSTEM MODEL

The system scheduling model is based on the seminal model in [16] and its extensions for FIFO queues [25] and mixed messages [21]. The system consists of a number of nodes connected to a single CAN network. A node may implement a priority queue or a FIFO queue. In the former case, the node is designated as a PQ-node and it enters the highest priority message from its transmit queue in the bus arbitration. Whereas, in the later case the node is identified as an FQ-node and it enters the oldest message from its transmit queue in the bus arbitration.

Each CAN message m_m has a unique identifier and a priority denoted by ID_m and P_m respectively. The priority of a message is assumed to be equal to its ID. The priority of the message m_m is considered higher than the priority of another message m_n if $P_m < P_n$. Let the sets $hp(m_m)$, $lp(m_m)$, and $hep(m_m)$ contain the messages with priorities higher, lower, and equal and higher than m_m respectively. Although the priorities of CAN messages are unique, the set $hep(m_m)$ is 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 ξ_m that denotes the transmission type of a message. ξ_m specifies whether m_m is periodic (P), sporadic (S) or mixed (M). Formally, the domain of ξ_m can be defined as follows.

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

Each message m_m has a transmission time C_m and queueing jitter J_m which is inherited from the task that queues m_m, i.e., the sending task. We assume that J_m can be smaller, equal or greater than T_m or MUT_m. Each message can carry a data payload that ranges from 0 to 8 bytes. This integer value is specified in the header field of the frame called Data Length Code and is denoted by s_m. In the case of periodic transmission, m_m has a transmission period which is denoted by T_m. Whereas, in the case of sporadic transmission, m_m has the MUT_m time. B_m denotes the blocking time of m_m which refers to the largest amount of time m_m has to wait for the transmission of a lower priority message.

If an FQ-node transmits the message m_m then the set of all messages transmitted by this node is defined by $M(m_m)$. The Lowest priority message in $M(m_m)$ is denoted by L_m. The sum of the transmission time of all the messages in $M(m_m)$ is identified by C_m^{SUM}. The transmission time of the shortest and longest messages in $M(m_m)$ are denoted by C_m^{MIN} and C_m^{MAX} respectively. f_m denotes the maximum buffering time between the instant the message m_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 [25].

We duplicate a message when its transmission type is mixed. Hence, each mixed message m_m is treated as two separate messages, i.e., one periodic and the other sporadic. The duplicates share all the attributes except for T_m and MUT_m. The periodic copy inherits T_m while the sporadic copy inherits the 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). m_m is deemed schedulable if its R_m is less than or equal to its deadline D_m. The system is considered schedulable if all of its messages are schedulable.

We consider the deadlines to be arbitrary which means that they can be greater than the periods or MUTs of corresponding messages. We assume that the CAN controllers are capable of buffering more than one instance of a message. The instances of a message are assumed to be transmitted in the same order in which they are queued (i.e., we assume FIFO policy among the instances of the same message). For better readability, all the notations used in this paper are tabulated at the end of the paper.

V. EXTENDED ANALYSIS

We extend the existing analysis of CAN with both PQ-nodes and FQ-nodes [25] by adapting the RTA of CAN for mixed messages [21]. Let the message under analysis be denoted by m_m. The extended analysis treats a message differently based on its transmission type. Here we consider two different cases. In the first case, m_m is assumed to be a periodic or a sporadic message. Whereas, m_m is considered to be a mixed message in the second case.

A. CASE 1: WHEN MESSAGE UNDER ANALYSIS IS PERIODIC OR SPORADIC

Consider m_m to be a periodic or a sporadic message. We calculate the worst-case response time of a message differently depending upon the type of the queueing policy implemented in the sending node. That is, we treat the message under analysis differently for the PQ-and FQ-nodes. Therefore, once again, we consider two cases: (a) the first case assumes that m belongs to a node that implements priority queue, (b) the second case considers that m belongs to a node that implements FIFO queue.

1) CASE 1 (A): WHEN MESSAGE UNDER ANALYSIS BELONGS TO A PRIORITY-QUEUED NODE

Since we consider arbitrary deadlines for messages, there can be more than one instance of m_m that may become ready for transmission before the end of priority level-m *maximum busy period*. The maximum busy period is the longest contiguous interval of time during which m_m is unable to complete its transmission due to two reasons. First, the network is occupied by the higher priority messages. In other words, at least one message of priority level-m or higher has not completed its transmission. Second, a lower priority message already started its transmission when m_m is queued for transmission. The maximum busy period starts at the so-called *critical instant*. In a system where messages are scheduled without offsets, the critical instant corresponds to the point in time when all higher priority messages in the system are queued simultaneously with m_m while their subsequent instances are queued after the shortest possible interval of time [18].

There can be another reason to check if more than one instance of m_m is queued for transmission in the priority level-m maximum busy period. Since, the message transmission in CAN is non-preemptive, the transmission of previous instance of m_m could delay the current instance of a higher priority message that may add to the interference received by the current instance of m_m. This phenomenon was identified by Davis et al. [18] and termed as ‘‘push-through interference’’. Because of this interference, a higher priority message may be waiting for its transmission before the transmission of the current instance of m_m finishes. Hence, the length of busy period may extend beyond T_m or MUT_m.

Intuitively, the response time of each instance of m_m within priority level-m maximum busy period should be calculated. The largest value among the calculated response times of all instances of m_m is considered as the worst-case response-time of m_m. Let q_m be the index variable to denote instances of m_m. The worst-case response time of m_m is given by:

$$R_m = max\{R_m(q_m)\} \tag{1}$$

Constituents of the worst-case response time. According to the existing analysis [16], [18], the worst-case response-time of any instance of m_m consists of three parts as follows.

1) The queueing jitter denoted by J_m. It is inherited from the sending task, i.e., the task that queues m_m for transmission. Basically, it represents the maximum variation in time between the release of the sending task and queuing of the message in the transmit queue (buffers). It is calculated by taking the difference between the worst- and best-case response time of the sending task.
2) The worst-case transmission time denoted by $\boldsymbol{C_m}$. It represents the longest time it takes for m_m to be transmitted over the network.
3) The queueing delay denoted by ω_m. It is equal to the longest time that elapses between the instant m_m is queued by the sending task in the transmit queue and the instant when m_m is about to start its successful transmission. In other words, ω_m is the interference caused by other messages to m_m.

Thus, the worst-case response time of any instance q_m of a periodic or sporadic message m_m is given by the following set of equations.

$$R_m(q_m) = \begin{cases} J_m + \omega_m(q_m) - q_m T_m + C_m, & \text{if } \xi_m = \text{P} \\ J_m + \omega_m(q_m) - q_m MUT_m + C_m, & \text{if } \xi_m = \text{S} \end{cases} \tag{2}$$

The terms $q_m T_m$ and $q_m MUT_m$ in (2) are used to support the response-time calculations for multiple instances of m_m. If the transmission type of m_m is periodic then the message period is taken into account. However, if the transmission type of m_m is sporadic, minimum update time is used in the above equation.

Calculations for the worst-case transmission time $\boldsymbol{C_m}$. The worst-case transmission time of m_m can be calculated using the method derived in [16] and later adapted in [18]. For the standard CAN identifier format, C_m is calculated as follows.

$$C_m = \left(47 + 8s_m + \left\lfloor \frac{34 + 8s_m - 1}{4} \right\rfloor\right)\tau_{bit} \tag{3}$$

Where τ_{bit} represents the time required to transmit a single bit of data on the CAN network. Its value depends upon the speed of the network. In (3), *47* is the number of bits due to protocol overhead. It is composed of start of frame bit (1-bit), arbitration field (12-bits), control field (6-bits), Cyclic Redundancy Check (CRC) field (16-bits), acknowledgement (ACK) field (2-bits), End of Frame (EoF) field (7-bits), and inter-frame space (3-bits). The number of bits due to protocol overhead in the case of extended CAN frame format is equal to *67*.

In [31], Broster identified that the analysis in [16] and [18] uses *47*-bits instead of *44*-bits as the protocol overhead. This is because the analysis in [16] and [18] accounts the 3-bit inter-frame space as part of the CAN frame. The 3-bit inter-frame space must be considered when calculating the interferences or blocking from other messages. However, Broster argued that this adds slight amount of pessimism to the response time of the message under analysis if the 3-bit inter-frame space is also considered in its transmission time. This is because the destination node can access the message before the inter-frame space. In order to avoid this pessimism, we subtract 3-bit time from the response time of the instance of the message under analysis.

The term $\left\lfloor \frac{34+8s_m-1}{4} \right\rfloor$ in (3) is added to compensate for the extra time due to *bit stuffing*. It should be noted that the bit sequences *000000* and *111111* are used for error signals in CAN. In order to be unambiguous in non-erroneous transmission, a stuff bit of opposite polarity is added whenever there are five bits of the same polarity in the sequence of bits to be transmitted [18]. The value *34* indicates that only 34-bits out of 47-bits protocol overhead are subjected to bit stuffing. The term $\lfloor \frac{a}{b} \rfloor$ is the notation for *floor* function. It returns the largest integer that is less than or equal to $\frac{a}{b}$.

For the message with extended CAN identifier format, C_m is calculated as follows.

$$C_m = \left(67 + 8s_m + \left\lfloor \frac{54 + 8s_m - 1}{4} \right\rfloor\right)\tau_{bit} \qquad (4)$$

The calculations for C_m in (3) can be simplified as follows.

$$C_m = (55 + 10s_m)\tau_{bit} \qquad (5)$$

Similarly, the calculations for C_m in (4) can be simplified as follows.

$$C_m = (80 + 10s_m)\tau_{bit} \qquad (6)$$

Calculations for the worst-case queueing delay ω_m. The calculations for ω_m should include the interference caused by all the other periodic, sporadic and mixed messages. The existing analyses for CAN with FIFO queues [22], [25], [26] have a limitation that they consider the effect of interference from only periodic and sporadic messages.

It is important to mention that CAN uses fixed-priority non-preemptive scheduling, 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_m has to wait in the transmit queue because the higher priority messages win the arbitration, i.e., the right to transmit before m_m. For a message queued at a PQ-node, ω_m is calculated by the following fixed-point iteration.

$$\omega_m^{n+1}(q_m) = B_m + q_m C_m + \sum_{\forall m_k \in hp(m_m)} I_k C_k \qquad (7)$$

The last term in (7) represents the interference from the higher priority messages. In order to solve this iterative equation, initial value of ω_m^n can be taken as follows.

$$\omega_m^0(q_m) = B_m + q_m C_m \qquad (8)$$

The iterations in (7) 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, the 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 the set of lower priority messages with respect to m_m and is given by the following equation.

$$B_m = \max_{\forall m_k \in lp(m_m)} (C_k) \qquad (9)$$

A higher priority message m_k contributes an extra delay, equal to f_k, to the worst-case queueing delay of m_m if m_k belongs to the FQ-node. f_k represents the delay after which the higher priority message m_k belonging to the FQ-node becomes the oldest message in the queue and can take part in the priority-based arbitration [25]. The existing analysis for mixed messages in CAN [21] does not take this additional delay into account. f_k is zero if m_k belongs to a PQ-node. We will come back to the calculations for f_k in Section V-C.

In (7), I_k is calculated differently for different values of ξ_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.

$$I_k = \begin{cases} \left\lceil \frac{\omega_m^n(q_m)+J_k+f_k+\tau_{bit}}{T_k} \right\rceil, & \text{if } \xi_k = \text{P} \\ \left\lceil \frac{\omega_m^n(q_m)+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{S} \\ \left\lceil \frac{\omega_m^n(q_m)+J_k+f_k+\tau_{bit}}{T_k} \right\rceil & \\ + \left\lceil \frac{\omega_m^n(q_m)+J_k+f_k+\tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{M} \end{cases} \qquad (10)$$

Length of the maximum busy period. The length of priority level-m maximum 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 calculated by the following iterative equation.

$$t_m^{n+1} = B_m + \sum_{\forall m_k \in hep(m_m)} I_k' C_k \qquad (11)$$

I_k' is given by the following relation. Note that the contribution of both the duplicates of a mixed message m_k in the set $hep(m_m)$ is taken into account.

$$I_k' = \begin{cases} \left\lceil \frac{t_m^n+J_k+f_k}{T_k} \right\rceil, & \text{if } \xi_k = \text{P} \\ \left\lceil \frac{t_m^n+J_k+f_k}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{S} \\ \left\lceil \frac{t_m^n+J_k+f_k}{T_k} \right\rceil + \left\lceil \frac{t_m^n+J_k+f_k}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{M} \end{cases} \qquad (12)$$

In order to solve the iterative equation (11), C_m can be used as the initial value of t_m^n as shown below.

$$t_m^0 = C_m \qquad (13)$$

The right hand side of (11) is a monotonic non-decreasing function of t_m. Equation (11) 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,

$$U_m < 1 \qquad (14)$$

where U_m is calculated by the following equation:

$$U_m = \sum_{\forall m_k \in hep(m_m)} C_k I''_k \tag{15}$$

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

$$I''_k = \begin{cases} \frac{1}{T_k}, & \text{if } \xi_k = \text{P} \\ \frac{1}{MUT_k}, & \text{if } \xi_k = \text{S} \\ \frac{1}{T_k} + \frac{1}{MUT_k}, & \text{if } \xi_k = \text{M} \end{cases} \tag{16}$$

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

The number of instances of m_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 for mixed messages).

$$Q_m = \begin{cases} \left\lceil \frac{t_m + J_m}{T_m} \right\rceil, & \text{if } \xi_m = \text{P} \\ \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil, & \text{if } \xi_m = \text{S} \end{cases} \tag{17}$$

The index of each message instance is identified by q_m and its range is given as follows.

$$0 \le q_m \le (Q_m - 1) \tag{18}$$

2) CASE 1 (B): WHEN MESSAGE UNDER ANALYSIS BELONGS TO A FIFO-QUEUED NODE

Similar to the existing RTA for CAN with FIFO queues [25], the extended analysis is FIFO-symmetric. This means that all the messages belonging to FQ-node will have same upper bound for their worst-case response times. In order to derive the worst-case response time of a periodic or sporadic message belonging to the FQ-node, we consider the worst-case conditions. Hence, we assume that the message under analysis is the lowest priority message, i.e., L_m in the group $M(m_m)$ with the largest transmission time C_m^{MAX} (to maximize the interference from the messages in $M(m_m)$ as well as from the messages belonging to other nodes). The response time of a particular instance q_m of a periodic or sporadic message m_m that is queued at the FQ-node is given by the following equation.

$$R_m(q_m) = \begin{cases} J_m + \omega_m(q_m) - q_m T_m + C_m^{MAX}, & \text{if } \xi_m = \text{P} \\ J_m + \omega_m(q_m) - q_m MUT_m + C_m^{MAX}, & \text{if } \xi_m = \text{S} \end{cases} \tag{19}$$

In [25], message deadlines are assumed to be equal to or less than the corresponding periods. Hence, for any message m_m belonging to $M(m_m)$ in the FQ-node, there could be only one instance of every other message queued ahead of m_m. In the existing analysis, the maximum amount of interference received by m_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_m in the FIFO queue. This is the reason we select the transmission time of m_m in FIFO-queued nodes to be equal to C_m^{MAX} instead of C_m^{MIN}.

Interference received by $\boldsymbol{m_m}$ from the messages in $\boldsymbol{M(m_m)}$. Now, we derive an upper bound for the number of instances of each message in the group $M(m_m)$ that can be queued ahead of m_m. Consider a simple but intuitive example as shown in Figure 6. Let the message under analysis be m_m (lowest priority message in $M(m_m)$). Also consider an arbitrary message m_i belonging to the group $M(m_m)$. Assume both m_i and m_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 6. 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 m_i queued ahead of any instance of m_m.

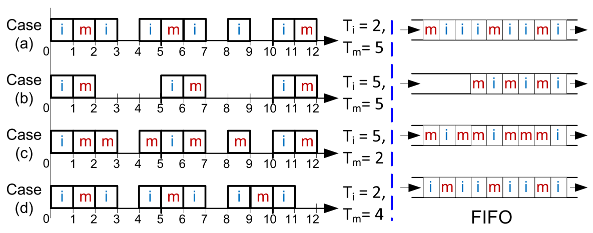

FIGURE 6. Demonstration of maximum interference on $\boldsymbol{m_m}$ from the messages in the group $\boldsymbol{M(m_m)}$.

The periods of m_i and m_m in each case are shown in Figure 6. The left hand side of Figure 6 shows the time line during which each instance of m_i and m_m is queued in the FIFO queue. Whereas, the right hand side of Figure 6 depicts the corresponding FIFO queue as if none of the messages was transmitted. The maximum number of instances of m_i that are queued ahead of any instance of m_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 m_i in the group $M(m_m)$ that can be queued ahead of any instance of m_m in the FIFO queue. We can generalize Q_i for all the cases as follows.

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

Let us consider the effect of jitter of m_i, denoted by J_i, on the interference of m_m. Because of J_i, additional instances of m_i can be queued ahead of m_m. Thus, taking the effect of jitter into account, (20) can be written as:

$$Q_i = \left\lceil \frac{T_m + J_i}{T_i} \right\rceil \tag{21}$$

Since, m_i can be periodic, sporadic or mixed, we can generalize (21) as follows.

$$Q_i = \begin{cases} \left\lceil \frac{T_m+J_i}{T_i} \right\rceil, & \text{if } \xi_i = \text{P} \\ \left\lceil \frac{T_m+J_i}{MUT_i} \right\rceil, & \text{if } \xi_i = \text{S} \\ \left\lceil \frac{T_m+J_i}{T_i} \right\rceil + \left\lceil \frac{T_m+J_i}{MUT_i} \right\rceil, & \text{if } \xi_i = \text{M} \end{cases} \tag{22}$$

Calculations for the worst-case queueing delay. The worst-case queueing delay, ω_m, in (19) can be calculated in a similar fashion as in (7) with the addition of extra delay shown in (22).

$$\omega_m^{n+1}(q_m) = B_{L_m} + \sum_{\forall m_i \in M(m_m) \wedge i \neq m} Q_i C_i + q_m C_m^{MAX} + \sum_{\forall m_k \in hp(L_m) \wedge m_k \notin M(m_m)} I_k C_k \tag{23}$$

Where m_k is any message that has priority higher than the lowest priority message in the FQ-node in which m_m is queued. Moreover, m_k does not belong to the FQ-node in which m_m is queued. m_i is any message, other than m_m, in the group $M(m_m)$. B_{L_m} is the blocking time of L_m which refers to the maximum transmission time of a message in the set of messages with lower priority than L_m that are sent by the other nodes. Since, the interference contributed to m_m by higher priority messages from other nodes (both PQ and FQ) is independent of m_m belonging to a PQ-node or FQ-node, I_k can be calculated using (10). The initial value of ω_m^n to solve the iterative equation (23) can be selected as follows.

$$\omega_m^0 = B_{L_m} + \sum_{\forall m_i \in M(m_m) \wedge i \neq m} Q_i C_i + q_m C_m^{MAX} \tag{24}$$

Length of the maximum busy period. The length of priority level-m maximum busy period, denoted by t_m, can be calculated in a similar fashion as in (11) and by following the intuition from (23). The effect of extra delay from the messages belonging to the FQ-nodes is also taken into account. t_m can be calculated by the following iterative equation.

$$t_m^{n+1} = B_{L_m} + \sum_{\forall m_i \in M(m_m) \wedge i \neq m} Q_i C_i + \sum_{\forall m_k \in hep(L_m) \wedge m_k \notin M(m_m)} I'_k C_k \tag{25}$$

The initial value for t_m^n can be selected using (13). Since, the interference to m_m by higher priority messages from other nodes (both PQ and FQ) is independent of m_m belonging to a PQ-node or FQ-node, I'_k can be calculated using (12). Similarly, the total number of instances of m_m that becomes ready for transmission before the busy period ends can be calculated using (17). The worst-case response time of m_m is the largest value of response time among all its instances as shown in (1).

B. CASE 2: WHEN MESSAGE UNDER ANALYSIS IS MIXED

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

$$R_m = max(R_{m_P}, \ R_{m_S}) \tag{26}$$

1) CASE 2 (A): WHEN MESSAGE UNDER ANALYSIS BELONGS TO A PRIORITY-QUEUED NODE

For a priority-queued mixed message, the response times of each instance of m_{m_P} and m_{m_S} are calculated separately by adapting the existing analysis for mixed messages in CAN [21]. Let us denote the total number of instances of m_{m_P} and m_{m_S}, occurring in the priority level-m maximum busy period, by Q_{m_P} and Q_{m_S} respectively. Assume that the index variable for message instances of m_{m_P} and m_{m_S} is denoted by q_{m_P} and q_{m_S} respectively. Their ranges are given by the following equations.

$$0 \leq q_{m_P} \leq (Q_{m_P} - 1) \tag{27}$$
$$0 \leq q_{m_S} \leq (Q_{m_S} - 1) \tag{28}$$

The worst-case response time of m_{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.

$$R_{m_P} = max(R_{m_P}(q_{m_P})) \tag{29}$$

Similarly, the worst-case response time of m_{m_S} 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.

$$R_{m_S} = max(R_{m_S}(q_{m_S})) \tag{30}$$

The worst-case response time of each instance of m_{m_P} and m_{m_S} can be derived by adapting the equations for the calculation of worst-case response time of periodic and sporadic messages respectively (derived in the first case) as given by the following two equations.

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P} T_m + C_m \tag{31}$$
$$R_{m_S}(q_{m_S}) = J_m + \omega_{m_S}(q_{m_S}) - q_{m_S} MUT_m + C_m \tag{32}$$

The queueing jitter, J_m, is the same (equal) in both the equations (31) and (32). The transmission time, C_m, is also the same in these equations and is calculated using (5) or (6) depending upon the type of CAN frame identifier. Although, both the duplicates of m_m inherit same J_m and C_m from it, they experience different amount of worst-case queueing delays caused by other messages.

Calculations for the worst-case queueing delay. The worst-case queueing delay experienced by m_{m_P} and m_{m_S} is denoted by ω_{m_P} and ω_{m_S} in (31) and (32) respectively. ω_{m_P} and ω_{m_S} can be calculated by adapting the equation for the calculations of worst-case queueing delay in (7). 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_{m_P} and m_{m_S} have equal priorities, any number of instances of m_{m_P} queued ahead of m_{m_S} contribute an extra delay to the worst-case queueing delay experienced by m_{m_S} and vice versa. We adapt the calculations for self interference in a mixed message that we derived in [21]. The worst-case queueing delay for m_{m_P} and m_{m_S} can be calculated using the following equations.

$$\omega_{m_P}^{n+1}(q_{m_P}) = B_m + q_{m_P}C_m + \sum_{\forall m_k \in hp(m_m)} I_{k_P}C_k + Q_{m_S}^P C_m \quad (33)$$

$$\omega_{m_S}^{n+1}(q_{m_S}) = B_m + q_{m_S}C_m + \sum_{\forall m_k \in hp(m_m)} I_{k_S}C_k + Q_{m_P}^S C_m \quad (34)$$

The effect of self interference can be seen in the last terms of (33) and (34). $Q_{m_S}^P$ denotes the total number of instances of m_{m_S} that are queued ahead of $q_{m_P}^{th}$ instance of m_{m_P}. Similarly, $Q_{m_P}^S$ denotes the total number of instances of m_{m_P} that are queued ahead of $q_{m_S}^{th}$ instance of m_{m_S}. The values of $Q_{m_S}^P$ and $Q_{m_P}^S$ are calculated as follows.

$$Q_{m_S}^P = \begin{cases} \left\lceil \frac{q_{m_P}T_m + J_m + \tau_{bit}}{MUT_m} \right\rceil, & \text{if } (q_{m_P} = 0)\ \&\&\ (J_m = 0) \\ \left\lceil \frac{q_{m_P}T_m + J_m}{MUT_m} \right\rceil, & \text{otherwise} \end{cases} \quad (35)$$

$$Q_{m_P}^S = \begin{cases} \left\lceil \frac{q_{m_S}MUT_m + J_m + \tau_{bit}}{T_m} \right\rceil, & \text{if } (q_{m_S} = 0)\ \&\&\ (J_m = 0) \\ \left\lceil \frac{q_{m_S}MUT_m + J_m}{T_m} \right\rceil, & \text{otherwise.} \end{cases} \quad (36)$$

In order to solve the iterative equations (33) and (34), the initial values of $\omega_{m_P}^n(q_{m_P})$ and $\omega_{m_S}^n(q_{m_S})$ can be selected according to (8) in a similar fashion. I_{k_P} and I_{k_S} are calculated using to the following equations.

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

$$I_{k_S} = \begin{cases} \left\lceil \frac{\omega_{m_S}^n(q_{m_S}) + J_k + f_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi_k = \text{P} \\ \left\lceil \frac{\omega_{m_S}^n(q_{m_S}) + J_k + f_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{S} \\ \left\lceil \frac{\omega_{m_S}^n(q_{m_S}) + J_k + f_k + \tau_{bit}}{T_k} \right\rceil + \\ \left\lceil \frac{\omega_{m_S}^n(q_{m_S}) + J_k + f_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi_k = \text{M} \end{cases} \quad (38)$$

The values of I_{k_P} and I_{k_S} in (37) and (38) differ from those calculated in [21] in a way that we consider an extra jitter, i.e., f_k from every message that belongs to the FQ-node.

Calculations for the length of the maximum busy period. The length of priority level-m maximum busy period, denoted by t_m, can be calculated using (11) that is developed for periodic and sporadic messages in a PQ-node. This is because (11) 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 this equation. Therefore, there is no need to calculate t_m for m_{m_P} and m_{m_S} separately. It should be calculated only once for a mixed message.

Although t_m is the same for m_{m_P} and m_{m_S}, the number of instances of both the messages that become ready for transmission just before the end of the maximum busy period, i.e., Q_{m_P} and Q_{m_S} respectively, may be different. The reason is that the calculations for Q_{m_P} and Q_{m_S} require T_m and MUT_m respectively and which may have different values. Q_{m_P} and Q_{m_S} can be calculated by adapting (17) that is derived for the calculation 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_S} are given by the following equations.

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

$$Q_{m_S} = \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil \quad (40)$$

2) CASE 2 (B): WHEN MESSAGE UNDER ANALYSIS BELONGS TO A FIFO-QUEUED NODE

The worst-case response times of each instance of m_{m_P} and m_{m_S} queued at the FQ-node are calculated similar to the case of FIFO-queued messages that are periodic or sporadic.

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P}T_m + C_m^{MAX} \quad (41)$$

$$R_{m_S}(q_{m_S}) = J_m + \omega_{m_S}(q_{m_S}) - q_{m_S}MUT_m + C_m^{MAX} \quad (42)$$

Calculations for the worst-case queueing delay. The worst-case queueing delays for m_{m_P} and m_{m_S} are calculated by adapting the calculations in the equations (23), (33) and

(34) as follows.

$$\omega_{m_P}^{n+1}(q_{m_P}) = B_{L_m} + \sum_{\forall m_i \in M(m_m) \wedge i \neq m} Q_i C_i + q_{m_P} C_m^{MAX} + \sum_{\forall m_k \in hp(L_m) \wedge m_k \notin M(m_m)} I_{k_P} C_k + Q_{m_S}^{P} C_m \quad (43)$$

$$\omega_{m_S}^{n+1}(q_{m_S}) = B_{L_m} + \sum_{\forall m_i \in M(m_m) \wedge i \neq m} Q_i C_i + q_{m_S} C_m^{MAX} + \sum_{\forall m_k \in hp(L_m) \wedge m_k \notin M(m_m)} I_{k_S} C_k + Q_{m_P}^{S} C_m \quad (44)$$

Since, the interference caused by higher priority messages from other PQ- and FQ-nodes is independent of the mixed message m_m belonging to a PQ-node or FQ-node, I_{k_P} and I_{k_S} can be calculated using (37) and (38). The initial values of ω_{m_P} and ω_{m_S} can be selected according to (24) while considering the respective index of each instance of m_{m_P} and m_{m_S}. The value of Q_i is calculated using (22) similar to the case of FIFO queued periodic or sporadic messages. The values of $Q_{m_S}^{P}$ and $Q_{m_P}^{S}$ are calculated using (35) and (36) that are derived for mixed message in a PQ-node. $Q_{m_S}^{P}$ denotes the total number of instances of m_S that are queued ahead of $q_{m_P}^{th}$ instance of m_P. Therefore, we consider only queueing jitter in (35) and do not take into account any additional delay that may occur after queueing of m_{m_P} such as f_m. Similar arguments hold for $Q_{m_P}^{S}$.

Calculations for the length of the maximum busy period. The length of priority level-m maximum busy period, denoted by t_m, can be calculated by using (25) that is developed for periodic and sporadic messages in a FQ-node. This is because (25) 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 the queueing delay from the duplicate is also covered in (25). Therefore, there is no need to calculate t_m for m_{m_P} and m_{m_S} separately. It should be calculated only once for a mixed message.

Although the length of the busy period is the same, the number of instances of m_{m_P} and m_{m_S} that become ready for transmission just before the end of the maximum busy period, i.e., Q_{m_P} and Q_{m_S} respectively, may be different. Q_{m_P} and Q_{m_S} can be calculated by following the same reasoning and using the equations that we derived for a mixed message in the PQ-node in (39) and (40) respectively.

C. ALGORITHM FOR THE CALCULATIONS OF MAXIMUM BUFFERING TIME IN FIFO QUEUES

The algorithm for the calculations of maximum buffering time in FIFO queues is adapted from [25] to support mixed messages in CAN with FIFO queues. The buffering time for any priority-queued message is equal to zero. It should be noted that the calculations for the response times in equations (2), (19), (31), (32), (41) and (42) are dependent upon the corresponding iterative calculations for the queueing delays in (7), (23), (33), (34), (43) and (44) respectively. Whereas, the calculations for queueing delay depends upon the maximum buffering time. Therefore, the response times and maximum buffering times should be calculated iteratively and simultaneously as shown in Algorithm 1.

Algorithm 1 Algorithm for the Calculations of Maximum Buffering Times and Message Response Times Simultaneously

```
1:  begin
2:  for all Messages in the system do
3:      $f_m \leftarrow 0$   ▷ Initialize buffering time for all messages.
4:  end for
5:  $Repeat \leftarrow TRUE$
6:  while $Repeat = TRUE$ do
7:      REPEAT ← FALSE
8:      for Every message $m_m$ in the system do
9:          if $m_m \in$ ECU with FIFO queue then
10:             if Message type of $m_m$ == PERIODIC or SPORADIC then
11:                 CALCULATE $R_m$ USING EQUATION (19)
12:             else if Message type of $m_m$ == MIXED then
13:                 CALCULATE $R_m$ USING EQUATIONS (41) AND (42)
14:             end if
15:             if $R_m > D_m$ then
16:                 $m_m$ IS UNSCHEDULABLE
17:             end if
18:             if $f_m < \omega_m$ then
19:                 $f_m \leftarrow \omega_m$
20:                 REPEAT ← TRUE
21:             end if
22:         else if $m_m \in$ ECU with priority queue then
23:             $f_m \leftarrow 0$   ▷ buffering time for a priority queued message is always zero.
24:             if Message type of $m_m$ == PERIODIC or SPORADIC then
25:                 CALCULATE $R_M$ USING EQUATION (2)
26:             else if Message type of $m_m$ == MIXED then
27:                 CALCULATE $R_M$ USING EQUATIONS (31) AND (32)
28:             end if
29:             if $R_m > D_m$ then
30:                 $m_m$ IS UNSCHEDULABLE
31:             end if
32:         end if
33:     end for
34: end while
35: end
```

VI. CASE STUDY AND EVALUATION

In this section, we conduct an automotive-application case study. Basically, we adapt the case study of the experimental vehicle that is analyzed for only periodic messages in [32]. We implemented[4] the extended analysis in a freely-available tool MPS-CAN Analyzer [27]. Using this tool,

[4]The discussion about the implementation in the tool is not in the scope of this paper.

P_m	ξ_m	s_m	T_m (us)	MUT_m (us)	R_m[Pio] (us)	R_m[FIFO] (us)
1	P	8	12500	0	540	32440
2	S	8	0	12500	810	39210
3	M	8	12500	12500	1080	34870
4	S	8	0	12500	1620	32980
5	S	8	0	50000	1890	35410
6	M	8	50000	50000	2160	35450
7	S	8	0	100000	2700	42860
8	S	8	0	20000	2970	35410
9	M	8	50000	50000	3240	33000
10	S	8	0	125000	3780	35330
11	S	8	0	25000	4050	42860
12	S	3	0	10000000	4220	43140
13	M	8	100000	100000	4490	42980
14	P	8	100000	0	5030	40020
15	M	8	100000	100000	5300	42980
16	M	8	100000	100000	5840	42980
17	S	8	0	100000	6380	33250
18	P	8	1000000	0	6650	33250
19	S	8	0	1000000	6920	40020
20	P	8	1000000	0	7190	35600
21	P	8	1000000	0	7460	33250
22	M	8	500000	500000	7730	35720
23	P	8	500000	0	8270	35600
24	S	8	0	500000	8540	43140
25	P	8	500000	0	8810	40020
26	P	8	100000	0	9080	35680
27	S	8	0	100000	9350	43140
28	P	8	100000	0	9620	35600
29	S	8	0	1000000	9890	43140
30	M	8	1000000	1000000	10160	33270
31	S	8	0	1000000	10700	33250
32	M	8	20000	20000	10970	35680
33	S	8	0	50000	11510	35600
34	M	8	500000	500000	11780	33270
35	P	8	20000	0	12320	33250
36	P	8	500000	0	12590	40020
37	P	8	20000	0	14210	33250
38	S	8	0	200000	14480	42860
39	P	8	20000	0	14750	43140
40	P	8	200000	0	15020	35600
41	P	8	1000000	0	15290	43140
42	P	8	100000	0	15560	35600
43	S	8	0	100000	15830	40020
44	P	8	100000	0	16100	42860
45	S	8	0	50000	16370	33250
46	P	8	50000	0	16640	40020
47	S	8	0	50000	16910	35600
48	M	8	50000	50000	17180	40160
49	S	8	0	1000000	17720	35600
50	P	8	1000000	0	17990	43140
51	S	8	0	1000000	18260	40020
52	P	8	1000000	0	18530	42860
53	M	8	128000	128000	18800	43260
54	S	8	0	128000	19340	35680
55	P	8	128000	0	19610	35600
56	M	8	1000000	1000000	19880	40160
57	S	8	0	250000	22040	40020
58	M	3	250000	250000	22210	43160
59	M	8	500000	500000	22650	40160
60	M	8	500000	500000	23190	35680
61	M	7	500000	500000	23710	33270
62	M	8	500000	500000	24230	35720
63	S	2	0	500000	24650	35720
64	M	8	1000000	1000000	24920	35720
65	P	8	1000000	0	27080	35680
66	M	8	1000000	1000000	27350	35680
67	P	8	1000000	0	27890	35680
68	P	8	1000000	0	28160	43140
69	P	6	1000000	0	28390	42860
70	S	8	0	2000000	28660	33250
71	S	8	0	2000000	28930	42860
72	P	8	2000000	0	29200	43140
73	M	8	2000000	2000000	29470	43260
74	M	8	2000000	2000000	30010	40160
75	S	8	0	2000000	30550	35680
76	P	8	2000000	0	30820	42860
77	M	8	2000000	2000000	31090	35680
78	M	2	2000000	2000000	31390	42860
79	M	1	50000	50000	31670	40040
80	M	2	1000000	1000000	31950	42860
81	M	2	2000000	2000000	32250	43140

FIGURE 7. Attributes and calculated response times of the periodic, sporadic and mixed messages in the experimental vehicle.

we compare and evaluate the response times of periodic, sporadic and mixed messages in the experimental vehicle using the extended analysis for mixed messages in CAN with FIFO queues and the existing analysis for CAN with priority queues.

A. EXPERIMENTAL SETUP

There are six ECUs in the experimental vehicle that are connected to a single CAN network. The selected speed for CAN is 500 Kbit/s. There are 81 CAN messages in the system; out of which 27 are periodic, 27 are sporadic, while the remaining 27 are mixed. All the attributes of these messages are shown in the table depicted in Figure 7. The attributes of each message are identified as follows. The priority, transmission type, number of data bytes in the message, transmission period, and minimum update time are represented by P_m, ξ_m, s_m, T_m, and MUT_m respectively. We assume, the smaller the value of the P_m parameter of a message, the higher its priority. Accordingly, the message with priority 1 is the highest priority message, whereas the message with priority 81 is the lowest priority message in the system. All timing parameters are in microseconds. We perform two sets of experiments. In the first set, all ECUs in the system implement priority queues. In the second set of experiments, all ECUs implement FIFO queues. In both sets of experiments, each ECU has 32 buffers in the transmit queue.

B. COMPARISON OF VARIOUS RESPONSE-TIME ANALYSES

In the first set of experiments, the response times of all messages are calculated using the existing response-time analysis for mixed, periodic and sporadic messages in CAN with priority queues [21]. The calculated response times are denoted by $R_m[Prio]$ in the table in Figure 7. On the other hand, in the second set of experiments,

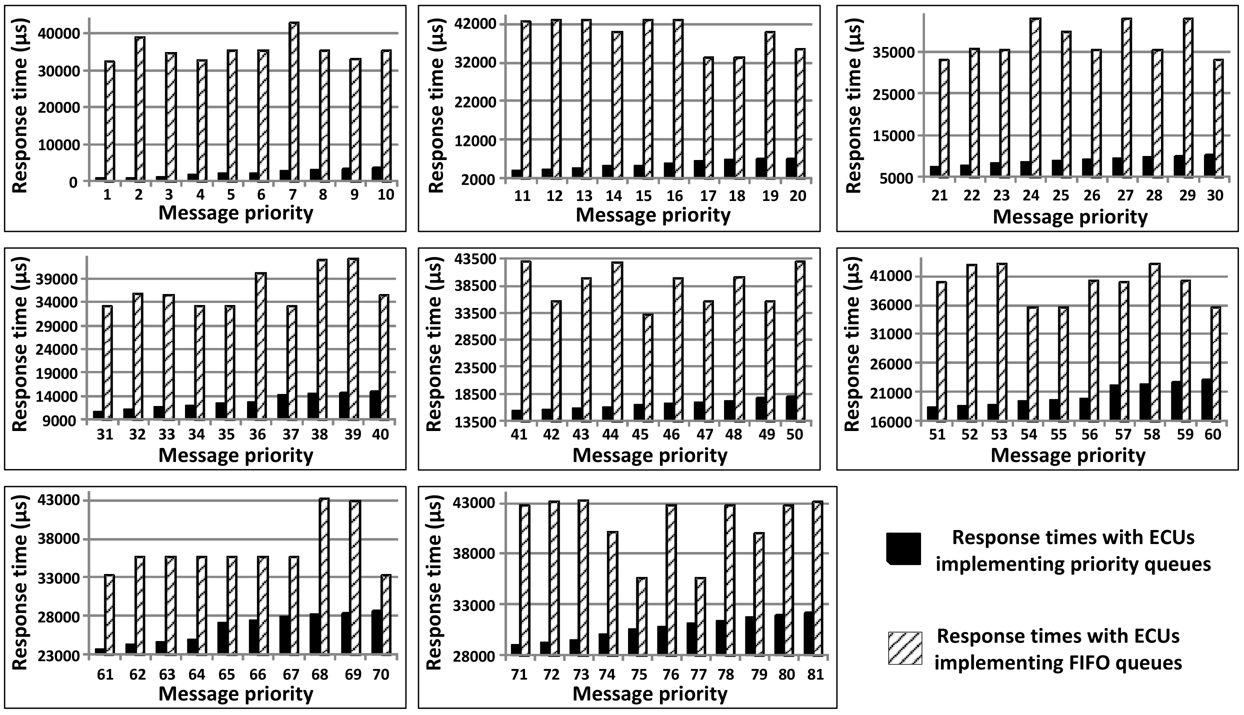

FIGURE 8. Comparison of message response-times with respect to different types of queueing policies in the ECUs.

the response times of all messages are calculated using the extended analysis presented in the previous section. The calculated response times in this case are denoted by $R_m[FIFO]$ in the table in Figure 7. The maximum network bandwidth utilization calculated in both cases is equal to 33.776970%.

The response times of all messages in these two cases are shown by the bar graphs in Figure 8. The first bar (solid black bar) in each set of the two bars represents the response time of a message in the system where all ECU's implement priority queues. Whereas, the second bar (pattern bar) in each set of the two bars represents the response time of a message in the system where all ECUs implement FIFO queues.

The response-time graphs show that the message response times are the best (smallest) in the case when all the ECUs use priority-based queueing policy. On the other hand, the response times of the messages are the worst (largest) in the system where the ECUs implement FIFO queues. In fact, the response times are significantly large in the case of FIFO queues. This is because of the priority inversion in FIFO queues (discussed in Section III-C). Moreover, the worst-case buffering time in the FIFO queues can be significantly large that adds to the worst-case response times of the messages.

C. DISCUSSION AND RECOMMENDATIONS

In order to get short response times of CAN messages, those ECUs should be selected that implement priority-based queueing policy. Although FIFO policy is simple and easy to implement, configure, and use as compared to the priority queueing policy, the messages can have very large worst-case response times in the case of ECUs implementing FIFO queues as shown in Figure 8. The ECUs which implement priority-based queueing policy should be preferred over the ECUs which implement FIFO queues especially in the systems that have high network utilization and short stimulus-to-response requirements. Moreover, it is important to use the right RTA that correctly matches the queueing policies in the ECUs; and transmission type of messages used in the higher-level protocols. If these constraints are not rightly considered in the RTA, the calculated response times can be optimistic.

VII. SUMMARY AND CONCLUSION

The existing worst-case response-time analysis for messages in Controller Area Network (CAN) with priority- and FIFO-queued nodes does not support the analysis of mixed messages. 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 based on CAN that are used in the automotive industry. We identified three different implementations of mixed messages in higher-level protocols for CAN. For some implementations, the existing analysis still provides safe upper bounds for worst-case response times. Whereas for the others, the existing analysis calculates optimistic worst-case response times.

We extended the existing analysis for 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 higher-level protocol for CAN that

APPENDIX A

TABLE 1. Notations and terminology.

Notation	Explanation
m_n	Any message $_n$
ID_n	Unique identifier of m_n
P_n	Priority of m_n
ξ_n	Transmission type of m_n. It specifies whether m_n is periodic (P), sporadic (S) or mixed (M)
C_n	Worst-case transmission time of m_n
J_n	Queueing jitter of m_n
s_n	Size of data payload in m_n
T_n	Transmission period of m_n
MUT_n	Minimum Update Time of m_n. It is the minimum time that should elapse between the transmission of any two sporadic messages
B_n	Blocking time of m_n
R_n	Worst-case response time of m_n
D_n	Deadline of m_n
$hp(m_n)$	Set of higher priority messages than m_n
$lp(m_n)$	Set of lower priority messages than m_n
$hep(m_n)$	Set of higher and equal priority messages than m_n
$lep(m_n)$	Set of lower and equal priority messages than m_n
ω_n	Queueing delay for m_n
f_n	Maximum buffering time for m_n
τ_{bit}	Time required to transmit a single bit of data over CAN
t_n	Length of the priority level-n busy period
q_n	Index variable to denote multiple instances of m_n
U_n	Bus utilization for priority level-n
Q_n	Total Number of instances of m_n that are queued in priority level-n busy period
$M(m_n)$	The set of FIFO-queued messages in the sender ECU of m_n
L_n	The lowest priority message in the set $M(m_n)$
B_{L_n}	Blocking time due to L_n
C_n^{MAX}	Maximum transmission time of a message in the set $M(m_n)$
C_n^{MIN}	Minimum transmission time of a message in the set $M(m_n)$
m_{n_P}	Periodic part of a mixed message m_n
m_{n_S}	Sporadic part of a mixed message m_n
R_{n_P}	Response time of m_{n_P}
R_{n_S}	Response time of m_{n_S}

supports periodic, sporadic, and mixed transmission of messages in a system comprising of priority- and FIFO-queued nodes. We conducted a case study and performed comparative evaluation of the extended analysis with the existing analysis for mixed, periodic and sporadic messages in CAN with priority queues.

The FIFO queues are already used in practical CAN controllers. Although, they are easy to implement and use, they can result in higher response times of messages. Therefore, the CAN controllers which implement priority queues should be preferred over the CAN controllers that implement FIFO queues. Moreover, it is important to use the response-time analysis that correctly matches the queueing policies in the ECUs; and transmission types of messages used in the higher-level protocols. If these constraints are not rightly considered in the response-time analysis, the calculated response times can be optimistic.

REFERENCES

[1] *CAN Specification Version 2.0*, Robert Bosch GmbH, Stuttgart, Germany, 1991.
[2] *Road Vehicles—Interchange of Digital Information—Controller Area Network (CAN) for High-Speed Communication*, ISO Standard 11898, Nov. 1993.
[3] *CAN With Flexible Data-Rate (CAN FD), White Paper, Ver. 1.1.*, Robert Bosch GmbH, Stuttgart, Germany, 2011.
[4] (2014, Feb. 5). *Automotive Networks. CAN in Automation (CiA)* [Online]. Available: http://www.can-cia.org/index.php?id=416
[5] P. Thorngren, "Keynote talk: Experiences from EAST-ADL use," in *Proc. EAST-ADL Open Workshop*, Oct. 2013.
[6] *CAL, CAN Application Layer for Industrial Applications*, CiA Draft Standard DS-207, Feb. 1996.
[7] (2014, Feb. 5). *CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Ver. 4.02.* [Online]. Available: http://www.can-cia.org/index.php?id=440
[8] *Hägglunds Controller Area Network (HCAN), Network Implementation Specification*, BAE Systems, Hägglunds, Sweden, Apr. 2009.
[9] (2014, Mar. 5). *MilCAN (CAN for Military Land Systems Domain)* [Online]. Available: http://www.milcan.org

[10] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Softw. Eng. J.*, vol. 8, no. 5, pp. 284–292, 1993.
[11] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: An historic perspective," *Real Time Syst.*, vol. 8, nos. 2–3, pp. 173–198, 1995.
[12] L. Sha *et al.*, "Real time scheduling theory: A historical perspective," *Real Time Syst.*, vol. 28, nos. 2–3, pp. 101–155, 2004.
[13] M. Joseph and P. Pandya, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, 1986.
[14] M. Nolin, J. Mäki-Turja, and K. Hänninen, "Achieving industrial strength timing predictions of embedded system behavior," in *Proc. Int. Conf. ESA*, 2008, pp. 173–178.
[15] 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 *Proc. 9th IEEE Int. WFCS*, May 2012, pp. 23–32.
[16] K. Tindell, H. Hansson, and A. Wellings, "Analysing real-time communications: Controller area network (CAN)," in *Proc. RTSS*, 1994, pp. 259–263.
[17] (2014, Feb. 5). *Volcano Network Architect. Mentor Graphics* [Online]. Available: http://www.mentor.com/products/vnd/communication-management/vna
[18] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, no. 3, pp. 239–272, 2007.
[19] (2014, Feb. 5). *Rubus-ICE: Integrated Component Development Environment* [Online]. Available: http://www.arcticus-systems.com
[20] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Comput. Sci. Inform. Syst.*, vol. 10, no. 1, pp. 453–482, 2013.
[21] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending schedulability analysis of controller area network (CAN) for mixed (periodic/sporadic)messages," in *Proc. 16th IEEE Conf. ETFA*, Sep. 2011, pp. 1–10.
[22] R. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Schedulability analysis for controller area network (CAN) with FIFO queues priority queues and gateways," *Real Time Syst.*, vol. 49, no. 1, pp. 73–116, 2013.
[23] D. Khan, R. Bril, and N. Navet, "Integrating hardware limitations in CAN schedulability analysis," in *Proc. 8th IEEE WFCS*, May 2010, pp. 207–210.
[24] M. D. Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. New York, NY, USA: Springer-Verlag, 2012.
[25] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller area network (CAN) schedulability analysis with FIFO queues," in *Proc. 23rd Euromicro Conf. Real Time Systems*, Jul. 2011, pp. 45–56.
[26] R. Davis and N. Navet, "Controller area network (CAN) schedulability analysis for messages with arbitrary deadlines in FIFO and work-conserving queues," in *Proc. 9th IEEE Int. WFCS*, May 2012, pp. 33–42.
[27] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Many-in-one response-time analyzer for controller area network," in *Proc. 4th Int. WATERS*, Jul. 2013.
[28] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogramming*, vol. 40, pp. 117–134, Apr. 1994.
[29] (2014, Feb. 5). *Requirements on Communication, Release 3.0, Rev 7, Ver. 2.2.0. The AUTOSAR Consortium* [Online]. Available: http://www.autosar.org/download/R3.0/AUTOSAR_SRS_COM.pdf
[30] (2013, Oct.). *AUTOSAR Techincal Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium* [Online]. Available: http://autosar.org
[31] I. Broster, "Flexibility in dependable real-time communication," Ph.D. dissertation, Dept. Comput. Sci., Univ. York, York, U.K., Aug. 2003.
[32] M. Di Natale and H. Zeng, "Practical issues with the timing analysis of the controller area network," in *Proc. 18th IEEE Conf. ETFA*, Sep. 2013, pp. 1–8.

SAAD MUBEEN is an Industrial Ph.D. Student with the Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden, and Arcticus Systems, Stockholm, Sweden. His research focus is on modeling and timing analysis of distributed real-time embedded systems in the automotive domain. He received the Licentiate degree in computer science and engineering from Mälardalen University in 2012, and the M.Sc. degree in electrical engineering with specialization in embedded systems from Jönköping University, Jönköping, Sweden, in 2009. He has co-authored over 40 research papers in peer-reviewed conferences, workshops, books, and journals.

JUKKA MÄKI-TURJA is a Senior Lecturer and Researcher with the Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden. His research interest lies in design and analysis of predictable real-time systems. He received the Ph.D. degree in computer science from Mälardalen University in 2005, with response time analysis for tasks with offsets as focus. He has co-authored over 75 research papers in peer-reviewed conferences, workshops, and journals.

MIKAEL SJÖDIN is a Professor of Real-Time System and a Research Director of Embedded Systems with Mälardalen University, Västerås, Sweden. His current research goal is to find methods that will make embedded-software development cheaper and faster, and to yield software with higher quality. Concurrently, he is also been pursuing research in analysis of real-time systems, where the goal is to find theoretical models for the real-time systems that will allow their timing behavior and memory consumption to be calculated. He received the Ph.D. degree in computer systems from Uppsala University, Uppsala, Sweden, in 2000. Since then, he has been involved in both academia and the industry with embedded systems, real-time systems, and embedded communications. Previous affiliations include Newline Information, Melody Interactive Solutions and CC Systems. In 2006, he joined the Mälardalen Real-Time Research Centre Faculty as a Full Professor with speciality in real-time systems and vehicular software-systems. He has co-authored over 200 research papers in peer-reviewed conferences, workshops, books, and journals.