This regular paper was presented as part of the main technical program at IEEE ETFA'2011

Extending Schedulability Analysis of Controller Area Network (CAN) for Mixed (Periodic/ Sporadic) Messages

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 schedulability analysis of Controller Area Network (CAN) developed by the research community is able to compute the response times of CAN messages that are queued for transmission periodically or sporadically. However, there are a few high-level protocols for CAN such as CANopen and Hägglunds Controller Area Network (HCAN) that support the transmission of mixed messages as well. A mixed message can be queued for transmission both periodically and sporadically. Thus, it does not exhibit a periodic activation pattern. The existing analysis of CAN does not support the analysis of mixed messages. We extend the existing analysis to compute the response times of mixed messages. The extended analysis is generally applicable to any high level protocol for CAN that uses any combination of periodic, event and mixed (periodic/ event) transmission of messages.

1. Introduction

Often, real-time systems are employed in distributed systems. In such systems, also known as distributed real-time systems, the nodes (processors) communicate with each other by sending and receiving messages over a real-time network or a bus. Controller Area Network (CAN) [10, 12] is a real-time, event-triggered, serial communication bus protocol. It supports bus speeds of up to 1 mega bits per second. CAN is a largely used real-time network in automotive domain. Moreover, it finds its application in other domains such as, medical equipments, industrial control, etc. There are many high level protocols and commercial extensions of CAN developed for many industrial applications. These include CAN Application Layer (CAL) [5], CANopen [6], Hägglunds Controller Area Network (HCAN) [22], CAN for Military Land Systems domain (MilCAN) [3], DeviceNet, etc.

System providers of hard real-time systems are required to ensure that the system meets its deadlines. Moreover, the need for safety criticality in most of the hard real-time systems requires an evidence that the actions by the system will be provided in a timely manner (e.g. each action will be taken at a time that is appropriate to the environment of the system). Therefore, it is important to predict the timing behavior of such systems. In order to provide the evidence that each action in the system will meet its deadline, *a priori* analysis techniques, also known as schedulability analysis techniques, have been developed by the research community.

Response-Time Analysis (RTA) [7, 18] 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 [15].

Tindell et al. [21] developed schedulability analysis of CAN which was recognized by the automotive industry. Later on, the analysis was revisited and revised by Davis et al. [9]. The model of communication used by this analysis assumes that the messages are queued for transmission by the application tasks which are activated periodically or sporadically. However, there are a few high-level protocols and commercial extensions of CAN such as CANopen and HCAN, that support the transmission of mixed messages as well. A mixed message contains both periodic and event signals. Thus a mixed message can be queued for transmission periodically as well as sporadically at the arrival of event signals. The current schedulability analysis of CAN does not support mixed messages.

In this paper, we extend the existing schedulability analysis of CAN to support the analysis of mixed messages. The extended analysis is able to find out the response times of periodic, event and mixed (periodic/event) CAN messages. The extended analysis is applicable to any high level protocol for CAN that uses any combination of periodic, event and mixed (periodic/event) transmission of

978-1-4577-0018-7/11/$26.00

messages. The motivation for this work comes from the activity of implementing the Holistic Response-Time Analysis (HRTA) [20] in the industrial tool suite, Rubus-ICE (Integrated Component development Environment) [1], that provides a component-based development environment for resource constrained distributed real-time systems.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we describe three different transmission patterns of CAN messages. In Section 4, we present the scheduling model for network communication. In Section 5, we visit the existing schedulability analysis of CAN and present the extended analysis. Finally, Section 6 concludes the paper.

2. Related Work

Liu and Layland [14] provided theoretical foundation for analysis of fixed-priority scheduled systems. Since then schedulability analysis of fixed-priority preemptive systems has been well developed. Joseph and Pandya published the first Response-Time Analysis (RTA) [13] for the simple task model presented by Liu and Layland which assumes independent periodic tasks.

There are many protocols such as CAN, TDMA (Time Division Multiple Access), TTCAN (Time-Triggered CAN), FlexRay, etc., that are used for real-time communication in distributed real-time systems. Schedulability analysis of these protocols has been developed by the research community. In this paper, we will focus only on the CAN protocol. Tindell et al. [21] developed the schedulability analysis of CAN by adapting the theory of fixed priority pre-emptive scheduling for uniprocessor systems. This analysis has been implemented in the analysis tools that are used in the automotive industry [4, 8]. Moreover, this analysis has served as basis for many research projects. Later on, this analysis was revisited and revised in [9]. The communication model used in this analysis supports the analysis of CAN messages that are queued for transmission periodically or sporadically. This analysis does not support the response-times computation of CAN messages that are queued for transmission both periodically and sporadically.

Tindell [20] developed the holistic schedulability analysis for distributed hard real-time systems. Holistic analysis combines both the schedulability analysis of nodes (uniprocessors) and the network. This analysis is able to analyze a distributed real-time system that employs CAN or a simple TDMA protocol. As discussed earlier, this analysis does not support the response-time computation of mixed type messages.

In [17], Pop et al. provide a holistic schedulability analysis of distributed embedded systems in which the tasks are both time- and event-triggered. The analysis is developed for ST/DYN protocol bus that uses static and dynamic phases for sending messages. Static phase is split into time slots and each node transmits in its own slot. The dynamic phase is shared by all nodes and the contention is resolved by message priorities. As compared to this approach, we use CAN protocol for network communication and the messages are queued by the tasks (that require remote transmission), on each node, periodically or sporadically or both periodically and sporadically.

3. Transmission Patterns of a CAN Message

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 [19]. 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 [21] 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 few high level protocols and commercial extensions of CAN in which the task that queues the messages can be invoked periodically as well as sporadically and hence, does not exhibit periodic activation patterns.

Throughout this paper, we will use the terms message and frame interchangeably since 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 frame as PERIODIC, EVENT or MIXED if it is queued by an application task that is invoked periodically, sporadically or both (periodically/ sporadically) respectively.

3.1. Periodic and Event Transmissions

If all the signals contained in a message are periodic then the transmission type of the message is periodic. Such a message will be queued for transmission at periodic intervals. On the other hand, if all the signals contained in a message are of event type then the message is said to have event transmission type. Such a message will be 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 event messages has elapsed. Hence, the transmission of an event frame is constrained by MUT.

3.2. Mixed (Periodic/Event) Transmission

If a message can be queued periodically as well as at the arrival of an event then the transmission type of a message is called mixed (periodic/event) or simply mixed transmission. We identified two different methods of implementing mixed messages for CAN protocol.

3.2.1 Method 1: Implementation of a Mixed Message

The CANopen protocol [2] provides an example of the first implementation method of a MIXED message. A mixed message can be queued for transmission at an arrival of

an event provided an 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 [2]. 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.

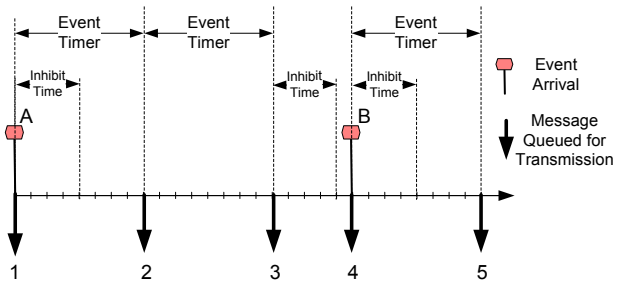

Figure 1. Transmission pattern of a Mixed Message in CANopen

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. Similarly, message 3 is queued for transmission because of 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 the message 4 is queued. The message 5 is transmitted because of the expiry of the Event Timer. Hence, there exist a dependency relationship between the Inhibit Time and the Event Timer.

3.2.2 Method 2: Implementation of a Mixed Message

The HCAN protocol [22] provides an example of the second implementation method of a MIXED message. A mixed message defined by HCAN protocol contains signals 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 event messages 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 2.

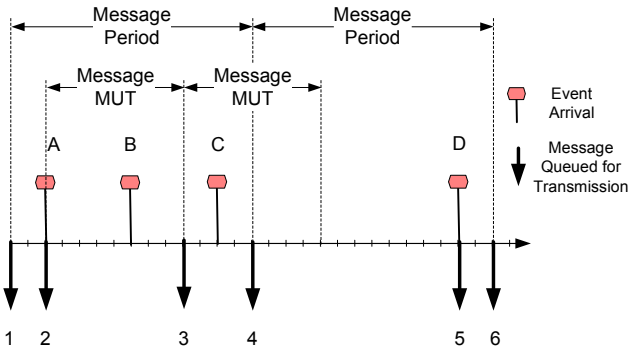

Figure 2. Transmission pattern of a Mixed Message in HCAN

In Figure 2, message 1 is queued for transmission because of the partly periodic nature of a mixed message. As soon as the event A arrives, message 2 is queued. When the 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 the 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 event message when MUT expires. It should be noted that the periodic transmission of a mixed message cannot be blocked by the event transmission. When an event D arrives, an event message 5 is immediately queued because the MUT has already expired. Message 6 is queued due to the periodicity.

3.2.3 Discussion

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 periodicity of transmission of a mixed message can never be higher than the Inhibit Time [6] [16]. Hence, it can be assumed that in the worst case, a mixed message is queued for transmission every time the Inhibit Timer expires. Therefore, the original CAN analysis can be used for mixed messages in the first method.

However, the second method of implementing a mixed message is more complex because the periodic transmission is independent of the event transmission. In other words, the Event Timer is not reset with every event transmission. In this case, for the purpose of analysis we need to treat a mixed message as two separate message streams with same IDs and priorities. This calls for the need of new analysis for mixed CAN messages. In addition, the existing analysis does not support any two messages with same

IDs and equal priorities, which also requires extension of the original analysis.

4. Network Scheduling Model

In this section, we discuss the network scheduling model that will be used in the development of extended analysis for mixed type CAN messages. This model is an extension to the communication model that was developed by Tindell et al. [21] for the response-time analysis of Controller Area Network (CAN) messages. The existing model supports the scheduling of messages that are queued for transmission periodically (PERIODIC messages) or sporadically (EVENT messages). We will extend this model to support the analysis of messages that are queued periodically as well as sporadically (MIXED messages).

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 EVENT or MIXED (both PERIODIC and EVENT). Each message has a unique priority (P_m), transmission time (C_m) and queueing jitter (J_m) which is inherited from the response time of the task queueing the message.

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 (DLC) and denoted by s_m. In case of PERIODIC transmission, each frame has a period, denoted by T_m. In case of EVENT transmission, each frame has a MUT_m that refers to the minimum time that should elapse between the transmission of any two EVENT 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).

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 separate messages. One copy is the PERIODIC message and the other is an EVENT message. 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 EVENT copy inherits MUT_m.

It is important to note that CAN identifier of each message is unique and it also corresponds to its priority. As discussed earlier that in case of a MIXED message, we duplicate the message and the duplicates have the same identifier and priority. The existing analysis model [19][21] does not support any two messages with equal priorities.

5. Extending CAN Schedulability Analysis

In this section, we extend the scheduling analysis of CAN that was originally developed by Tindell et al. [21] and later revised by Davis et al. [9]. The extended analysis will be able to compute the response times of mixed type messages as well.

5.1. Existing Analysis

First of all, we quickly revisit the existing algorithms that are used to compute the response-times of CAN messages. Then we extend these algorithms to support the analysis of mixed type messages.

According to the existing analysis, the worst-case response time of a CAN message is given by the following equation:

$$R_m = J_m + \omega_m + C_m \tag{1}$$

where m is the message under analysis. J_m denotes the queueing jitter of m and is inherited from the worst-case response time of the task that queues this message (sending task). ω_m represents the worst-case queueing delay and is equal to the longest time that elapses between the instant a message m is queued by the sending task in the priority-ordered send queue and the instant when the message starts its transmission. In other words, ω_m is the interference caused by other messages to m.

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 the message has to wait in the send queue because the higher priority messages win the arbitration and hence, the right of transmission before the message under analysis.

ω_m is given by the following recursive equation:

$$\omega_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{\omega_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{2}$$

In (2), $hp(m)$ refers to the set of all messages in the system that have higher priority than m. τ_{bit} denotes the time required to transmit a single bit on CAN bus. Its value depends upon the speed of the bus. In order to solve the recursive equation given by (2), initial value of ω_m^n can be taken equal to the blocking time, B_m, as given by the following equation:

$$\omega_m^0 = B_m \tag{3}$$

B_m represents the maximum time for which m can be blocked by the lower priority messages. It is equal to the largest transmission time of any message in the set of all

the lower priority messages compared to the priority of m and is given by the following equation:

$$B_m = \max_{\forall k \in lp(m)} (C_k) \tag{4}$$

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

In (2), C_m is the transmission time of m. It represents the longest time it takes for m to be transmitted over the bus. The transmission time of the message is computed according to [9] as given by the following equation:

$$C_m = \left(g + 8s_m + 13 + \left\lfloor \frac{g + 8s_m - 1}{4} \right\rfloor \right) \tau_{bit} \tag{5}$$

where s_m is the Data Length Code. It refers to the number of data bytes in a CAN data message. It can have any integer value from 0 to 8. g is equal to 34 and 54 for standard and extended CAN frame formats respectively. For a Standard CAN identifier, (5) can be simplified as follows.

$$C_m = (55 + 10s_m)\tau_{bit} \tag{6}$$

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

$$C_m = (80 + 10s_m)\tau_{bit} \tag{7}$$

In [9], Davis et al. made an observation that it is possible in the case of fixed-priority non-preemptive scheduling that a higher priority task may be waiting for transmission when a message m finishes its transmission. Hence, they proposed to analyze all the instances of m that lie in the level-m busy period.

In order to calculate the worst-case response time of a CAN message, 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 from the response time of all instances should be picked up as the worst-case response time of m. The length of a priority level-m busy period, t_m, is given by the following recursive equation:

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \tag{8}$$

where, $hep(m)$ refers to the set of all messages in the system that have equal or higher priority than m. In order to solve this recursive equation, initial value of t_m^n can be taken equal to the transmission time of m, i.e.

$$t_m^0 = C_m \tag{9}$$

The right hand side of (8) is a monotonic non-decreasing function of t_m. The recursive equation (8) is guaranteed to converge if the bus utilization for messages of priority level m and higher, denoted by U_m, is less than 1. U_m is given by the following equation:

$$U_m = \sum_{\forall k \in hep(m)} \frac{C_k}{T_k} \tag{10}$$

thus,

$$U_m < 1 \tag{11}$$

The number of instances of m, denoted by Q_m, that becomes ready for transmission before the end of the busy period is given by the following equation:

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{12}$$

The response time of each instance of m is calculated by the following equation:

$$R_m(q) = J_m + \omega_m(q) - qT_m + C_m \tag{13}$$

where q is the message-instance number. The range of q is shown below.

$$0 \leq q \leq Q_m - 1 \tag{14}$$

The queueing delay of each instant of the message m is given by the following equation.

$$\omega_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{\omega_m^n(q) + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{15}$$

After the response time of all instances of the message m have been computed, its worst-case response time can be found by selecting the largest value as given by the following equation.

$$R_m = max(R_m(q)), \qquad \forall\, 0 \leq q \leq (Q_m - 1) \tag{16}$$

5.2. Extended Analysis

In the extended schedulability analysis of CAN, 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. It can be either periodic or event or mixed. Formally, the domain of this function can be defined as:

$$\xi(m) \in [\text{PERIODIC}, \quad \text{EVENT}, \quad \text{MIXED}]$$

We assume that there are multiple slots for sending and receiving messages in the CAN controllers. Usually each slot has a single buffer [9]. If the previous instance of a message is not sent before the next then the previous instance is overwritten by the next one. In case of multiple buffers per slot, we assume that the FiFo (First in First out) policy is used to send the multiple instances of a message.

We discuss two cases. In the first, we assume that a message under analysis has a transmission type either periodic or event. Whereas in the second case, we consider that the message under analysis is of mixed transmission type.

5.2.1 Case 1: When the Message Under Analysis is Periodic or Event

When the transmission type of m is PERIODIC or EVENT then the worst-case response time of each instance q of this message is computed by the following equation:

$$R_m(q) = \begin{cases} J_m + \omega_m(q) - qT_m + C_m, \\ \qquad \text{if } \xi(\text{k}) = \text{PERIODIC} \\ J_m + \omega_m(q) - q(MUT_m) + C_m, \\ \qquad \text{if } \xi(\text{k}) = \text{EVENT} \end{cases} \tag{17}$$

This equation is similar to the response-time equation (13) in the existing analysis. In (17), J_m represents the queueing jitter which is equal to the worst-case response time of the task that queues m. C_m represents the transmission time of m. It is calculated according to the existing analysis using (6) or (7) depending upon the type of CAN frame identifier. If the transmission type of a message under analysis is PERIODIC then the message period is taken into account. However, if the transmission type of the message is EVENT, minimum update time is used in the above response-time equation.

The algorithms for the computation of the worst-case queueing delay (ω_m) of m should include the interference caused by all the other PERIODIC, EVENT and MIXED messages. The existing analysis accounts the interference caused by only PERIODIC and EVENT messages.

As we discussed in the communication model that when transmission type of a message is MIXED, we duplicate the message and designate the duplicates as a PERIODIC and EVENT copy of the MIXED message. It is important to note that all the attributes of the duplicates are the same as that of the original MIXED message except the PERIODIC copy inherits the period while the EVENT copy inherits minimum update time.

Worst Case Queueing Delay of a Periodic or Event Message

Each higher priority MIXED message should contribute more interference to the the message under analysis. The worst-case queueing delay, adapted from (15) in the existing analysis, can be computed by the following recursive equation:

$$\omega_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} I_k C_k \tag{18}$$

where I_k is computed differently for different values of $\xi(k)$ (k is the index of any higher priority message) as shown below. Note that the interference by a higher priority MIXED message contains the contribution from both the duplicates.

$$I_k = \begin{cases} \left\lceil \frac{\omega_m^n(q) + J_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{PERIODIC} \\ \left\lceil \frac{\omega_m^n(q) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{EVENT} \\ \left\lceil \frac{\omega_m^n(q) + J_k + \tau_{bit}}{T_k} \right\rceil + & \\ \quad \left\lceil \frac{\omega_m^n(q) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{MIXED} \end{cases} \tag{19}$$

The initial value of ω_m^n can be taken equal to the blocking time of m as given by (3). B_m in (18) can be computed by the same method which is used in the existing analysis given by (4). This is because CAN uses fixed priority non-preemptive scheduling and any message can be blocked by only one message in the set of lower priority messages. Although we duplicate all the mixed messages, a message under analysis can only be blocked by either the periodic copy or the event 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, event and mixed messages in a set of lower priority messages with respect to the message under analysis.

Length of the Busy Period

The length of priority level-m busy period, denoted by t_m, is also adapted from the existing analysis as given in (8). It can be computed by the following recursive equation.

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} I'_k C_k \tag{20}$$

where I'_k is given by the following relation. Note that the contribution of both the duplicates of a MIXED message k is taken into account, provided k belongs to a set of equal or higher priority messages with respect to m.

$$I'_k = \begin{cases} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{PERIODIC} \\ \left\lceil \frac{t_m^n + J_k}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{EVENT} \\ \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil + \left\lceil \frac{t_m^n + J_k}{MUT_k} \right\rceil, & \text{if } \xi(\text{k}) = \text{MIXED} \end{cases} \tag{21}$$

In order to solve this recursive equation, C_m can be used as an initial value of t_m^n as shown in (9). The right hand

side of (20) is a monotonic non-decreasing function of t_m. The recursive equation (20) 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,

$$U_m < 1 \tag{22}$$

where U_m is computed by the following equation:

$$U_m = \sum_{\forall k \in hep(m)} C_k I''_k \tag{23}$$

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

$$I''_k = \begin{cases} \frac{1}{T_k}, & \text{if } \xi(k) = \text{PERIODIC} \\ \frac{1}{MUT_k}, & \text{if } \xi(k) = \text{EVENT} \\ \frac{1}{T_k} + \frac{1}{MUT_k}, & \text{if } \xi(k) = \text{MIXED} \end{cases} \tag{24}$$

In the above equation, the contribution by both the copies of all the mixed messages, lying in a set of equal and higher priority messages with respect to m, is clearly 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):

$$Q_m = \begin{cases} \left\lceil \frac{t_m + J_m}{T_m} \right\rceil, & \text{if } \xi(m) = \text{PERIODIC} \\ \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil, & \text{if } \xi(m) = \text{EVENT} \end{cases} \tag{25}$$

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

$$0 \leq q \leq Q_m - 1 \tag{26}$$

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

$$R_m = max(R_m(q)), \qquad \forall\, 0 \leq q \leq (Q_m - 1) \tag{27}$$

5.2.2 Case 2: When the Message Under Analysis is Mixed

Since, a message with a MIXED transmission type is duplicated, we compute the response time of both the duplicates separately. For simplicity, we denote the PERIODIC and EVENT 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:

$$R_m = max(R_{m_P},\ \ R_{m_E}) \tag{28}$$

where, R_{m_P} and R_{m_E} are computed separately by adapting the existing analysis. Let us denote the total number of instances of messages 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:

$$0 \leq q_{m_P} \leq (Q_{m_P} - 1) \tag{29}$$

similarly,

$$0 \leq q_{m_E} \leq (Q_{m_E} - 1) \tag{30}$$

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

$$R_{m_P} = max(R_{m_P}(q_{m_P})) \tag{31}$$

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

$$R_{m_E} = max(R_{m_E}(q_{m_E})) \tag{32}$$

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 EVENT messages respectively, derived in case 1, as given by the following two equations:

$$R_{m_P}(q_{m_P}) = J_m + \omega_{m_P}(q_{m_P}) - q_{m_P} T_m + C_m \tag{33}$$

$$R_{m_E}(q_{m_E}) = J_m + \omega_{m_E}(q_{m_E}) - q_{m_E} MUT_m + C_m \tag{34}$$

The queueing jitter, J_m, is the same in both the equations (33) and (34). It is equal to the worst-case response time of the task that queues m. The transmission time, C_m, is also the same in these equations and is calculated according to the existing analysis by using (6) or (7) 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.

The worst-case queueing delay experienced by m_P and m_E is denoted by ω_{m_P} and ω_{m_E} in (33) and (34) respectively. ω_{m_P} and ω_{m_E} can be computed by adapting the algorithm for the computation of the worst-case queueing delay for PERIODIC and EVENT messages presented in (18). In this algorithm, we need to add the contribution of m_P to the worst-case queueing delay experienced by m_E and vice versa. It should be noted that the copies of a mixed message have equal priority and the existing analysis does not allow any two messages with equal priority.

Effect of Self Interference in a Mixed Message

In order to derive the contribution of one copy of a mixed message to the worst-case queueing delay of the other, consider three different cases, depicting the transmission pattern of a mixed message m, shown in Figure 3. In the first case, we assume that T_m is greater than MUT_m. This means that there could be more transmissions of the event copy compared to the periodic copy of m. Since the maximum update time between the queueing of any two event copies can be arbitrarily very long, it is also possible that there are fewer event transmissions than the periodic transmissions of m. In the second case, we assume that T_m is equal to MUT_m. In this case, there could be equal transmissions of both the copies of m. In the third case, we assume that T_m is smaller than MUT_m. This implies that the event transmissions will be less than the periodic transmissions of m.

It is important to note that in the example shown in Figure 3, there is a small offset between the first periodic and event transmission of m. This offset is used to maximize the queueing delay. If this offset is removed then only one frame will be queued corresponding to the first instance of both periodic and event copy. Moreover, the larger value between T_m and MUT_m is the integer multiple of the smaller in all the cases. This relationship along with the offset between T_m and MUT_m ensures that periodic and event transmission of m will not overlap, there by, maximizing the queueing delay.

Case (a): $\mathbf{T_m > MUT_m}$

Let the message under analysis be m_P and consider case (a) in Figure 3. An application task queues m periodically with a period T_m (e.g., equal to 9 time units). Moreover, the same task can also queue m at the arrival of events (labeled with numbers 1-6). The queueing of m_E is constrained by MUT_m (e.g., equal to 3 time units). The first instance of m_P, i.e., ($q_{m_P} = 0$), is queued for transmission as shown by $m_P(0)$ in Figure 3. If event 1 had arrived at the same time as the queueing of $m_P(0)$ then the signals in $m_E(0)$ were updated as part of $m_P(0)$. In that case, $m_E(0)$ was not queued separately (this is the property of a mixed message). In order to maximize the contribution of m_E on the queueing delay of m_P, $m_E(0)$ is queued just after the queueing of $m_P(0)$ as shown in all the cases in Figure 3. Therefore, $m_E(0)$ and subsequent instances of m_E will have no contribution in the worst-case queueing delay of the first instance of m_P, i.e., $m_P(0)$.

Now, consider the second instance of m_P. All the instances of m_E that are queued just before the queueing of $m_P(1)$ will contribute to its worst-case queueing delay. It can be observed in the case (a) that the first three instances of m_E are queued before $m_P(1)$. Similarly, there are six instances of m_E that are queued before $m_P(2)$.

Let $Q^P_{m_E}$ denotes the total number of instances of m_E that are queued before the $q^{th}_{m_P}$ instance of m_P. We can

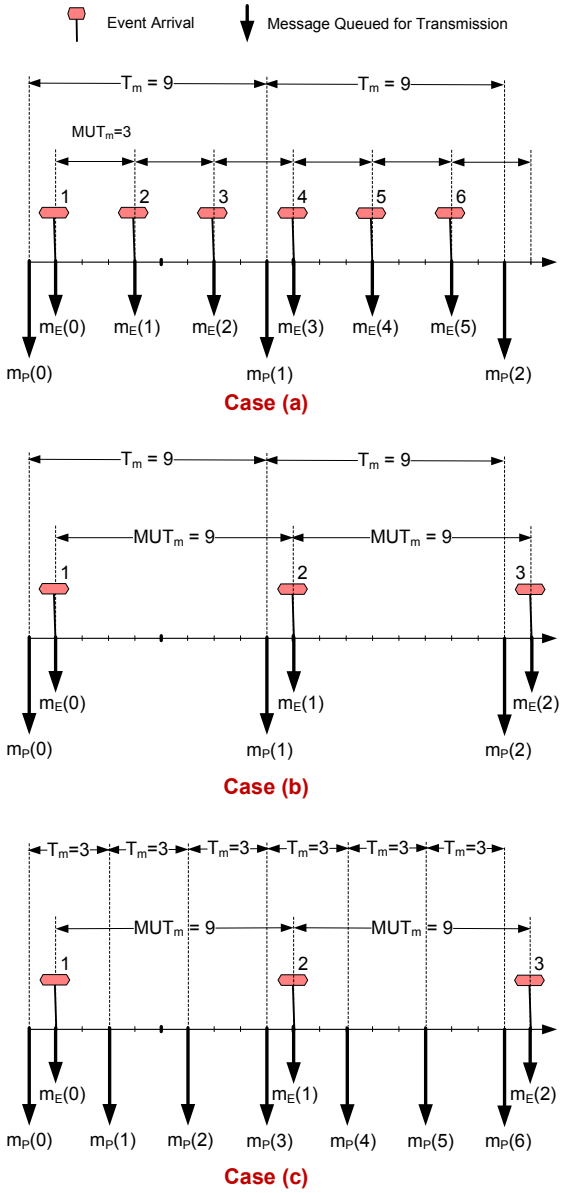

Figure 3. Demonstration of self interference in a MIXED message. Case (a) $\mathbf{T_m > MUT_m}$. Case (b) $\mathbf{T_m = MUT_m}$. Case (c) $\mathbf{T_m < MUT_m}$

generalize $Q^P_{m_E}$ for the case (a) as follows:

$$Q^P_{m_E} = \left\lceil \frac{q_{m_P} T_m}{MUT_m} \right\rceil \tag{35}$$

for example, consider again the queueing of different instances of m_E and m_P in the case (a). Equation (35) yields the set $\{Q^P_{m_E} = 0, 3, 6, ...\}$ for the corresponding values in the set $\{q_{m_P} = 0, 1, 2, ...\}$. Thus the total number of instances of m_E queued before each instance of m_p computed by (35) are consistent with the case (a) in Figure 3.

Case (b): $\mathbf{T_m = MUT_m}$

Consider case (b) in which T_m is equal to MUT_m. It

can be observed from Figure 3 that there are 0, 1, and 2 instances of m_E that are queued before $m_P(0)$, $m_P(1)$ and $m_P(2)$ respectively. When Equation (35) is used in case (b), we get the set $\{Q^P_{m_E} = 0, 1, 2, ...\}$ for the corresponding values in the set $\{q_{m_P} = 0, 1, 2, ...\}$. Therefore, (35) is also applicable on case (b).

Case (c): $\mathbf{T_m < MUT_m}$

Now, consider case (c) in which T_m (equal to 3 time units) is smaller than MUT_m (equal to 9 time units). The first instance of m_E, which is $m_E(0)$, will be queued before the queueing of $m_P(1)$, $m_P(2)$ and $m_P(3)$. Similarly, it can be seen from the figure that two instances of m_E, which are $m_E(0)$ and $m_E(1)$, will contribute to the worst-case queueing delay of $m_P(4)$, $m_P(5)$ and $m_P(6)$. (35) yields the set $\{Q^P_{m_E} = 0, 1, 1, 1, 2, 2, 2, ...\}$ for the corresponding values in the set $\{q_{m_P} = 0, 1, 2, 3, 4, 5, 6, ...\}$. Thus the total number of instances of m_E queued before each instance of m_p computed by Equation (35) are consistent with the case (c) in Figure 3.

Now we consider the effect of jitter on the instances of m_E previous to $m_E(0)$ which can be queued just before $m_P(0)$ and hence, can contribute to the worst-case queueing delay of m_P. We assume a FiFo queue for the queueing of different instances of each message. By adding the jitter of m_E to $Q^P_{m_E}$, equation (35) can be generalized for the three cases as follows.

$$Q^P_{m_E} = \left\lceil \frac{q_{m_P} T_m + J_m}{MUT_m} \right\rceil \tag{36}$$

The total number of instances of m_P that are queued before the $q^{th}_{m_E}$ instance of m_E, denoted by $Q^E_{m_P}$, can be derived in a similar fashion. Thus $Q^E_{m_P}$ can be computed by the following equation:

$$Q^E_{m_P} = \left\lceil \frac{q_{m_E} MUT_m + J_m}{T_m} \right\rceil \tag{37}$$

Worst Case Queueing Delay of a Mixed Message

The worst-case queueing delay of messages m_P and m_E can be computed by adapting (18) as follows.

$$\omega^{n+1}_{m_P}(q_{m_P}) = B_m + q_{m_P} C_m + \sum_{\forall k \in hp(m)} I_{k_P} C_k + Q^P_{m_E} C_m \tag{38}$$

$$\omega^{n+1}_{m_E}(q_{m_E}) = B_m + q_{m_E} C_m + \sum_{\forall k \in hp(m)} I_{k_E} C_k + Q^E_{m_P} C_m \tag{39}$$

Where, I_{k_P} and I_{k_E} are given by the following equations.

$$I_{k_P} = \begin{cases} \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + J_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(k) = \text{PERIODIC} \\ \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = \text{EVENT} \\ \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + J_k + \tau_{bit}}{T_k} \right\rceil + & \\ \left\lceil \frac{\omega^n_{m_P}(q_{m_P}) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = \text{MIXED} \end{cases} \tag{40}$$

$$I_{k_E} = \begin{cases} \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + J_k + \tau_{bit}}{T_k} \right\rceil, & \text{if } \xi(k) = \text{PERIODIC} \\ \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = \text{EVENT} \\ \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + J_k + \tau_{bit}}{T_k} \right\rceil + & \\ \left\lceil \frac{\omega^n_{m_E}(q_{m_E}) + J_k + \tau_{bit}}{MUT_k} \right\rceil, & \text{if } \xi(k) = \text{MIXED} \end{cases} \tag{41}$$

B_m in equations 38 and 39 can be computed by the same method which is used in the existing analysis given by (4). By using the values of $Q^P_{m_E}$ and $Q^E_{m_P}$ from (36) and (37) in equations (38) and (39), we get:

$$\omega^{n+1}_{m_P}(q_{m_P}) = B_m + q_{m_P} C_m + \sum_{\forall k \in hp(m)} I_{k_P} C_k + \left\lceil \frac{q_{m_P} T_m + J_m}{MUT_m} \right\rceil C_m \tag{42}$$

$$\omega^{n+1}_{m_E}(q_{m_E}) = B_m + q_{m_E} C_m + \sum_{\forall k \in hp(m)} I_{k_E} C_k + \left\lceil \frac{q_{m_E} MUT_m + J_m}{T_m} \right\rceil C_m \tag{43}$$

In order to solve the recursive equations (40) and (41), initial values of $\omega^n_{m_P}(q_{m_P})$ and $\omega^n_{m_E}(q_{m_E})$ can be taken equal to the blocking time of the MIXED message m, i.e.

$$\omega^0_{m_P}(q_{m_P}) = \omega^0_{m_E}(q_{m_E}) = B_m \tag{44}$$

Length of the Busy Period

The length of priority level-m busy period, denoted by t_m, can be computed by using (20) that was developed for PERIODIC and EVENT messages. This is because (20) 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 (20). Therefore, there is no need to compute t_m for m_P and m_E separately. t_m should be computed only once for a MIXED message m.

Although the length of the busy period is the same for m_P and m_E, the number of instances of both the messages

that become ready for transmission just before the end of busy period, i.e., Q_{m_P} and Q_{m_E} respectively, may be different. 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 (25) that was derived for the computation of the number of instances of PERIODIC and EVENT messages that become ready for transmission before end of the busy period. Q_{m_P} and Q_{m_E} are given by the following equations.

$$Q_{m_P} = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{45}$$

$$Q_{m_E} = \left\lceil \frac{t_m + J_m}{MUT_m} \right\rceil \tag{46}$$

6. Conclusion

The schedulability analysis of Controller Area Network (CAN) developed by the research community can compute the response times of CAN messages that are queued by application tasks periodically or sporadically. The existing analysis does not support the analysis of mixed messages. A mixed message can be queued for transmission both periodically and sporadically. Mixed messages are used in some of the high-level protocols for CAN such as CANopen and HCAN. Hence, the context of this problem is very general and requires a new analysis to support mixed messages.

In this paper, we extended the existing schedulability analysis of CAN to support the analysis of mixed messages. The extended analysis is able to compute the response times of CAN messages with all types of transmission patterns, i.e., periodic, event and mixed. The extended analysis is applicable to any high level protocol or commercial extension of CAN that uses any combination of periodic, event and mixed (periodic/event) transmission of messages.

In future work, the extended analysis will be implemented in an existing industrial tool suite, the Rubus-ICE [11], that provides a complete component-based development environment for resource-constrained distributed real-time systems.

Acknowledgement

This work is supported by Swedish Knowledge Foundation (KKS) within the project 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 for the cooperation.

References

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

[2] CANopen Application Layer and Communication Profile. CiA Draft Standard 301. Version 4.02. February 13, 2002. www.ece.unh.edu/biolab/hof/public/CiA.

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

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

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

[6] CANopen high-level protocol for CAN-bus, Version 3.0. *NIKHEF, Amsterdam*, March 2000.

[7] 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.

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

[9] 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.

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

[11] K. Hänninen, J. Mäki-Turja, S. Sandberg, J. Lundbäck, M. Lindberg, M. Nolin, and K.-L. Lundbäck. Framework for real-time analysis in Rubus-ICE. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 782 –788, 2008.

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

[13] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal (British Computer Society)*, 29(5):390–395, October 1986.

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

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

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

[17] T. Pop, P. Eles, and Z. Peng. Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems. In *Proceedings of the tenth international symposium on Hardware/software codesign*, CODES '02, pages 187–192, New York, NY, USA, 2002. ACM.

[18] 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.

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

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

[21] 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.

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