CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Schedulability analysis for CAN bus messages of periodically-varying size

Ishfaq Hussain*
Pedro Souto*
Konstantinos Bletsas*
Muhammad Ali Awan
Eduardo Tovar*

*CISTER Research Centre

CISTER-TR-220501

2022/04/27

Schedulability analysis for CAN bus messages of periodically-varying size

Ishfaq Hussain*, Pedro Souto*, Konstantinos Bletsas*, Muhammad Ali Awan, Eduardo Tovar*

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: hussa@isep.ipp.pt, pfs@fe.up.pt, ksbs@isep.ipp.pt, awa@isep.ipp.pt, emt@isep.ipp.pt

https://www.cister-labs.pt

Abstract

Abstract 14Conventional CAN bus schedulability analysis assumes that all messages with a given identifier have the same worst-case length. In this paper we extend that analysis to amore general model in which messages with a given identifier may have different lengths, that vary according to a known periodic pattern. That is, for some positive integer S, we assume that the length of message instances n and n + S with the same id is the same. By leveraging such patterns, where present, our new analysis allows for a more efficient use of CAN bus bandwidth than the application of conventional analysis, which can be pessimistic. This may be interesting when a given node sends the values of multiple signals with different periods. In such a scenario, the conventional CAN schedulability analysiswould require either the use of different ids for different signals (assuming there are enough of them), which leads to a higher bandwidth overhead because of the reduplication of message headers, or using only one id, but pessimistically always assuming the maximum possible length of the message, for safety reasons.

Schedulability analysis for CAN bus messages of periodically-varying size

Ishfaq Hussain*, Pedro F. Souto†, Konstantinos Bletsas*, Muhammad Ali Awan*, Eduardo Tovar*
*CISTER Research Centre and ISEP/IPP, Porto, Portugal
†University of Porto, FEUP-Faculty of Engineering and CISTER Research Centre, Porto, Portugal

2022 IEEE 18th International Conference on Factory Communication Systems (WFCS) | 978-1-6654-1086-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/WFCS53837.2022.9779187

***Abstract*—Conventional CAN bus schedulability analysis assumes that all messages with a given identifier have the same worst-case length. In this paper we extend that analysis to a more general model in which messages with a given identifier may have different lengths, that vary according to a known periodic pattern. That is, for some positive integer S, we assume that the length of message instances n and $n + S$ with the same id is the same. By leveraging such patterns, where present, our new analysis allows for a more efficient use of CAN bus bandwidth than the application of conventional analysis, which can be pessimistic. This may be interesting when a given node sends the values of multiple signals with different periods. In such a scenario, the conventional CAN schedulability analysis would require either the use of different ids for different signals (assuming there are enough of them), which leads to a higher bandwidth overhead because of the reduplication of message headers, or using only one id, but pessimistically always assuming the maximum possible length of the message, for safety reasons.**

I. Introduction

The CAN bus ("Controller Area Network") is a communication bus architecture, standardized as ISO 11898, that is ubiquitous in automotive systems. The original motivation behind CAN was to replace many kilometers of physical wire connecting the ever-more-numerous sensors and electronic control units with a lightweight and robust logical interconnect. There currently exist more than a billion (10^9) automotive systems with CAN buses in them [1], so the real-world importance of this technology is very high.

One of the main characteristics of CAN bus is that, by design, it supports *at the physical layer* the prioritized transmission of different messages according to their distinguishing message identifier (which doubles as a priority level). Without getting into too much detail, there exist a contention round and a transmission round. During the *contention round*, all nodes currently contending for the bus transmit their corresponding message identifiers in binary representation while also monitoring the bus to see what bit is transmitted. Any node that reads from the bus a binary value not matching what it is currently transmitting, "knows" that it does not have the highest-priority among the contenders, and should give up and retry at the next contention round. This scheme is properly called Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) [2]. At the end of the contention round (typically 11 bits long), the highest-priority current contender "knows" that it can transmit its message (typically, 0 to 8 bytes, before any bit-stuffing) during the subsequent **transmission round**, without any contention on the bus. All other nodes can then receive that message (although, typically, they will only care for specific message ids).

This property of CAN proved very valuable because it can be leveraged to achieve very high (typically 80% or more [1]) bandwidth utilization with offline-derivable guarantees of timely message delivery. Indeed, Tindell et al. noted the analogies with uniprocessor fixed-priority scheduling and came up with schedulability analysis for the CAN bus [3]–[5]. The state-of-the-art form of that analysis by Davis et al. [2] further influenced other works on that topic [6]–[10]. Before this theoretical toolkit was available, researchers had to resort to lengthy simulations or test runs to have some confidence of timeliness (but no hard guarantees!), and the typical bus utilization was only up to 30% [1].

In this paper, we aim to remove the pessimism that occurs when the (worst-case) length of a given CAN message is not fixed (as the state-of-the-art analysis assumes), but instead varies for successive instances of the message, according to a known periodically-repeating pattern. More formally, this is the case where for some integer $S(> 1)$, the worst-case length of message n and $n + S$ with the same identifier is the same. This generalization of the CAN message model is analogous to the generalization of the worst-case execution times (WCETs) of real-time computing tasks[1], devised by Mok and Chen [11]. Note that our contributions are purely at the analytical level, and do not involve any changes at all to CAN itself or its implementations. After all, CAN does not restrict messages of a given id to always have the same length.

Our work is motivated by practical considerations. For example, consider a CAN node which sends the values of multiple signals with different periods that are multiple of each other (Figure 1). The conventional CAN schedulability analysis can deal with this if different message ids are used for different signals, but this has the disadvantage of higher bandwidth overhead, because of the reduplication of message headers and the elevated message contention. Additionally, there might not always exist enough available message ids, so they would need to be used sparingly. Alternatively, the same

[1]This generalization is known as the "multiframe task model". The term "frame" therein carries specific meaning, not to be confused with the concept of transmission frames in communication protocols, such as CAN itself.

message (and a single id) can be used to piggy-back multiple sensor values, with the message length varying accordingly. However, the current analysis will then have to conservatively always assume the maximum possible message length, leading to pessimism.

Our approach, by comparison, allows for using a single message id, in such cases, without any of the analytical pessimism that the state-of-the-art analysis incurs.

The rest of this paper is composed of the following sections. In Section II, background about CAN schedulability analysis and multiframe task model is presented. The system model is discussed in Section III. Section IV presents a relatively quick but pessimistic analysis for multisized CAN messages. Tighter analysis is developed in Section V. Section VI illustrates, via examples, the application of the two new analyses and the kind of improvement they can achieve over the state-of-the art. Finally, conclusions are drawn in Section VII.

II. Background

Our analysis builds on the worst-case response time analysis of the CAN protocol by Davis et al. [2] and on the response time analysis for the multiframe task model by Baruah et al. [12]. In this section, we summarize those existing results.

A. CAN Schedulability Analyis

The state-of-the-art in CAN schedulability analysis is found in [2], superseding all earlier seminal works [3]–[5] and fixing certain issues. In [2], Davis et al. apply uniprocessor fixed-priority based response time analysis to the schedulability of the CAN network. Indeed, we can map the problem of scheduling messages on a bus to the problem of scheduling task on a single core processor. The main difference between these two problems arises from the fact that conventional fixed-priority-based response time analysis assumes preemptive tasks, whereas, in CAN, a message whose transmission has already began cannot be preempted by a higher-priority message that arrived meanwhile.

More specifically, the analysis by Davis et al. [2] bounds the time since the arrival of a message, m, until the end of its transmission. Each message is assumed to have a worst-case transmission time, C, a minimal inter-arrival time, T, and a deadline for its transmission, D, relative to its arrival. The worst-case response time (WCRT) of message m is given by:

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

where

- J_m is the queuing jitter, the longest time between the arrival of a message and adding it to the transmission queue;
- w_m is the queuing delay, the longest time the message remains in the transmission queue, until its successful transmission on the bus begins;
- C_m is the longest message transmission time.

The queuing delay has two components:

- B_m is the longest time m may have to wait for a lower-priority message to complete its transmission. This models the case when such a message is being transmitted when m is queued.
- I_m is the worst-case interference caused by higher priority messages that arrive before m starts being transmitted and therefore win access to the bus.

The blocking time is given by:

$$B_m = \max_{i \in lp(m)} (C_i) \tag{2}$$

where $lp(m)$ is the set of messages of lower priority than m.

The computation of I_m relies on the concept of *busy period*, introduced by Lehoczky [13].

Davis et al. define a priority level-m busy period as a time interval $[t^s, t^e)$, i.e. a right-open interval, where:

- t^s is the time when a message of priority m or higher is queued for transmission, and there are no messages of priority m or higher waiting to be transmitted that were queued strictly before t^s;
- t^e is the earliest time when the bus becomes idle, i.e. ready for transmission, and there are no messages of priority m or higher waiting to be transmitted that were queued strictly before time t^e.

Note that, during a level-m busy period, messages with priority lower than m do not start transmission. Furthermore, all messages with priority higher than or equal to m that are queued during a busy period are transmitted during that busy period.

Thus the largest length, t_m, of a level-m busy period is given by the recurrence:

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \tag{3}$$

where $hep(m)$ is the set of messages with priority higher than or equal to m.

This recurrence is guaranteed to converge provided:

$$\sum_{\forall k \in hep(m)} \frac{C_k}{T_k} < 1 \tag{4}$$

i.e., the total bus utilization of messages with priority higher than or equal to m is less than 1.

The number of instances Q_m of message m that arrive during a level-m busy period is given by:

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{5}$$

Thus, the analysis computes worst-case response time of each of these instances.

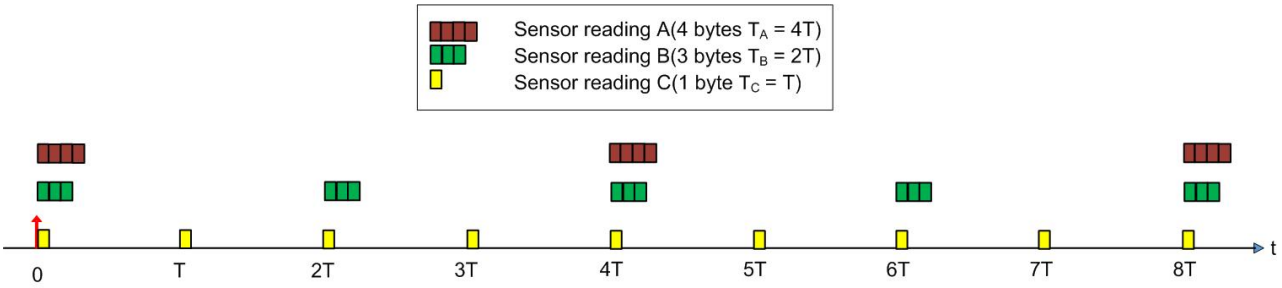

Fig. 1: The transmission of different sensor readings with periods that are multiple of each other can be combined on the same message; the size of subsequent instances of that message will accordingly vary.

Let q denote the order of an instance of a message m in a level-m busy period. Thus $q = 0$ for the first instance and $q = Q_m - 1$ for the last one.

The longest time from the start of the level-m busy period until instance q begins successful transmission is given by:

$$w_m^{n+1}(q) = B_m + q\,C_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{6}$$

where τ_{bit} is a CAN bit transmission time.

The response time of instance q is given by:

$$R_m(q) = J_m + w_m(q) - q\,T_m + C_m \tag{7}$$

Thus, the worst-case response time of message m is:

$$R_m = \max_{q=0..Q_m-1}(R_m(q)) \tag{8}$$

Note that if for some q, we have $R_m(q) > D_m$ the message is unschedulable, and the analysis should stop.

Finally, the worst-case transmission time C_m of a message m (which includes the transmission of its identifier and bits added for error-resilience and synchronization, including bit-stuffing) as a function of its size s_m in bytes is given by [2]:

$$C_m = (55 + 10 \cdot s_m) \cdot \tau_{bit} \qquad \text{if ids are 11-bit} \tag{9}$$

$$C_m = (80 + 10 \cdot s_m) \cdot \tau_{bit} \qquad \text{if ids are 29-bit} \tag{10}$$

B. Multiframe Task Model

The multiframe task model was proposed by Mok and Chen [11], in order to reduce the pessimism in the schedulability analysis of systems with computing tasks whose worst-case execution times vary according to a known periodic pattern. Specifically, the WCET of consecutive jobs of a multiframe task τ_i follow a pattern that repeats every F_i jobs. That is, in any schedule, the k^{th}, $(k + F_i)^{th}$, $(k + 2F_i)^{th}$, ... jobs of τ_i all have the same WCET. Therefore, each multiframe task $\tau_i = ((C_{i,0}, C_{i,1}, \ldots, C_{i,(F_i-1)}), T_i)$ is characterized by a vector of WCET $(C_{i,0}, C_{i,1}, \ldots, C_{i,(F_i-1)})$ and minimum inter-arrival time T_i. (In [11], the term "frame" refers to each of the F_i elements of that vector.)

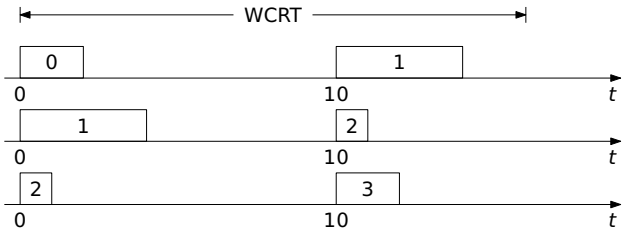

Fig. 2: Interference of a multiframe task depends on the phasing of its jobs w.r.t. the job under analysis, whose WCRT is shown. (The rectangles represent the jobs of the interfering task, and the number inside each of them is the job number.)

To determine the schedulability of a multiframe task, it suffices to check if the WCRT of the frame with the largest WCET is smaller than the task deadline (assumed to be constrained, i.e., $D_i \leq T_i$). However, unlike the "single-frame" model, it is not enough to compute the number of job releases of each interfering task in the WCRT of the task under analysis. This is because each multiframe task τ_i is characterized by a vector of WCET $\left(C_{i,0}, C_{i,1}, \ldots, C_{i,(F_i-1)}\right)$, and the phasing of the released jobs affects the amount of interference. This is illustrated in Figure 2, which shows the 3 possible phasings of jobs of an interfering task τ_i with period $T_i = 10$ and WCET vector $(2, 4, 1)$, w.r.t. to a job under analysis whose WCRT is 16 time units. During this interval there are at most 2 jobs of the interfering task, but the amount of interference depends on the first job in that sequence. As illustrated, this interference is worst, 6, if the first job in the sequence is job 0, F_k, ... On the other hand, if the first job in the sequence is job 2, $2 + F_k$, ..., the interference is the least, 3.

To efficiently compute the interference exerted by a multiframe task, Baruah et al. [12] define the $g(\tau_i, k)$ function:

$$g(\tau_i,k)=\begin{cases}0 & \text{if } k=0\\ \max\limits_{0\le j<F_i}\left(\sum\limits_{\ell=j}^{j+k-1}C_{i,(\ell \bmod F_i)}\right) & \text{if } 1\le k\le F_i\\ q\cdot g(\tau_i,F_i)+g(\tau_i,r) & \text{otherwise}\\ \quad \text{where } q=k \text{ div } F_i \text{ and} & \\ \quad\quad r=k \bmod F_i & \end{cases} \tag{11}$$

which bounds the cumulative WCET of any sequence of k jobs of task τ_i. This function is then used to define function $G(\tau_i,t)$:

$$G(\tau_i,t)=g\left(\tau_i,\left\lceil\frac{t}{T_k}\right\rceil\right) \tag{12}$$

which bounds the cumulative execution request of task τ_i over any time interval of duration t. Finally, the worst-case response time recurrence is:

$$R_i=g(\tau_i,1)+\sum_{\tau_j\in hp(i)}G\left(\tau_j,R_i\right) \tag{13}$$

III. System model

The assumed system model, other than capturing the possibility of a message's length varying according to a known cyclic pattern, does not deviate at all from the one assumed in [2]. The assumptions regarding the underlying CAN protocol and implementation are exactly the same. Specifically:

What is common with the state-of-the-art: We assume a number of nodes on the bus and a set of messages transmitted, with a known respective inter-arrival time T_m and relative deadline D_m for each message m. Message identifiers are unique and also serve as their priorities. Their size is 11 or 29 bits (standard/extended format, respectively). At the start of each CAN frame, nodes with messages ready for transmission, first transmit their respective identifiers, in the contention round. At the end of the contention round, only the winner proceeds with the transmission of the payload of its message, after subjecting to *bit-stuffing*. The payload of a message is 0 to 8 bytes long[2], before bit-stuffing – and bit-stuffing means the insertion of a complementary bit, whenever there is a sequence of 5 identical consecutive bits [3]. The bit length of a message after bit-stuffing therefore depends on the actual bit values, which may only be known at run time, but an upper bound is a function of the worst-case message length prior to bit-stuffing, which is known offline. In this paper, as in the literature, we therefore use the symbol C to denote the worst-case transmission time (CAN bit time) of a message. This contains the transmission of its identifier plus its payload (assuming maximum bit-stuffing).

[2]A zero-byte payload message can be an alarm of sorts, conveyed solely by the message identifier.

[3]This facilitates synchronization, since the nodes do not share a common clock in reality, and provides some error resilience. A detailed discussion of bit-stuffing would be out-of-scope for our paper, but there is some discussion in [2] for the interested reader.

multiframe task	multisized CAN message
job	message instance
number of frames	message cycle length
superframe	message cycle
frame	message subtype

TABLE I: Terminology for multiframe real-time tasks (left) and analogous concepts for multisized CAN messages (right).

The end of the transmission of a message is identified by all nodes upon detection of a *bus idle period*, which involves specific bit patterns Then, the next CAN frame starts. Any messages that became ready for transmission during the previous frame, will now compete in the new contention round.

Depending on whether or not CAN nodes have buffers capable of holding multiple ready untransmitted messages, message deadlines can either be constrained (i.e., $D_m \le T_m,\ \forall m$) or arbitrary (i.e,, without such restriction). This is because multiple untransmitted instances of a message with a worst-case response time $R_m > T_m$ may accumulate at the transmitting node.

What is new in this work: The state-of-the-art characterizes all message instances with a given id m, by the same worst-case transmission length C_m. This is pessimistic if the message length varies according to a known cyclic pattern. We call those *multisized messages*. Consider the following definitions pertaining to them, which mirror concepts from multiframe tasks (Table I).

The **message cycle length** S_m, characterizing a message m (and denoted as simply S, for ease of notation, whenever possible), is the least integer such that the worst-case transmission lengths of the k^{th} instance and the $(k+S)^{th}$ instance of message m are the same, for any k.

A **message cycle** consists of any consecutive S instances of message m.

Additionally, we call each of the S messages in a cycle, a **message subtype**.

IV. Analysis

A. Multisized messages

In order to upper-bound the interference caused by higher priority messages we adapt the $g(.)$ and $G(.)$ functions of multiframe tasks to multisized messages as follows:

$$g_m(k)=\begin{cases}0 & \text{if } k=0\\ \max\limits_{0\le j<S}\left(\sum\limits_{\ell=j}^{j+k-1}C_{m,(\ell \bmod S)}\right) & \text{if } 1\le k\le S\\ q\cdot g_m(S)+g_m(r) & \text{otherwise}\\ \quad \text{where } q=k \text{ div } S, \text{ and} & \\ \quad\quad r=k \bmod S & \end{cases} \tag{14}$$

where $C_{m,0} \dots C_{m,(S-1)}$ are the transmission times for each message in a sequence of S messages, and S is the *cycle length* for message m, i.e., the smallest integer such that $C_{m,k} = C_{m,(k+S)}$ for any integer k.

Similarly, we define the $G_m(t)$ function:

$$G_m(t) = g_m\left(\left\lceil \frac{t}{T_m} \right\rceil\right) \tag{15}$$

which upper-bounds the cumulative transmission request of message m within a time interval of duration t.

Consider a multisized message with a cycle length of 3 and a period of 200 τ_{bit} , and let its message subtypes have payloads of 2, 4 and 1 byte, respectively. Their corresponding worst-case transmission lengths (assuming 11-bit ids and applying Equation 9 would be 75, 95 and 65 τ_{bit}. Thus we have $g_m(1) = \max(75, 95, 65) = 95$, $g_m(2) = \max(75 + 95, 95 + 65, 65 + 75) = 170$ and $g_m(3) = \max(75 + 95 + 65, 95 + 65 + 75, 65 + 75 + 95) = 235$ τ_{bit}. And, over a time interval of length, e.g., 380 τ_{bit}, corresponding to the WCRT of some lower-priority message, the maximum cumulative transmission request (analogous to cumulative task execution request, in real-time task scheduling analysis) of message m is given by:

$$G_m(380) = g_m\left(\left\lceil \frac{380}{200} \right\rceil\right) = g_m(2) = 170 \ \tau_{bit}$$

In the rest of the paper, we will generally use τ_{bit} as the time unit, for ease of notation, without loss of generality.

B. Response time analysis

With these definitions, we are ready to generalize the analysis in Davis et al. [2] to multisized messages.

The analysis of the response time of message m starts with the computation of the length of the level-m busy period. For this computation rather than using (3), we use the following recurrence:

$$t_m^{n+1} = B_m + \sum_{\forall k \in hep(m)} G_k(t_m^n + J_k) \tag{16}$$

where:

$$B_m = \max_{i \in lp(m)} g_i(1) \tag{17}$$

A possible starting value is $t_m^0 = B_m + g_m(1)$.

Indeed, when different subtypes of a message may have different lengths, the blocking term can be as large as the maximum transmission time of the largest message subtype of each of the messages with lower priority than m. Furthermore, it is not enough to count the number of releases in the interval under consideration, rather we need also to consider the phasing of these instances. Since the right-hand-side of (16) is monotonically non-decreasing with respect to t_m, the convergence condition is similar to (4):

$$\sum_{\forall k \in hep(m)} U_k < 1 \tag{18}$$

where U_k is defined as follows:

$$U_k = \frac{1}{S} \sum_{j=0}^{S-1} \frac{C_{k,j}}{T_k} \tag{19}$$

where S is the message cycle length of message k.

The next steps are to compute the response time for each of the:

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil \tag{20}$$

instances of message m during the maximum length level-m busy period.

The longest time from the start of the level-m busy period until instance q, which ranges from 0 to $Q_m - 1$, begins successful transmission is given by:

$$w_m^{n+1}(q) = B_m + g_m(q) + \sum_{\forall k \in hp(m)} G_k(w_m^n(q) + J_k + \tau_{bit}) \tag{21}$$

Possible starting values are $w_m^0(q) = B_m$, for $q = 0$ and $w_m^0(q) = w_m(q-1) + g_m(q) - g_m(q-1)$, otherwise.

The response time of instance q is given by:

$$R_m(q) = J_m + w_m(q) - q\,T_m + g_m(1) \tag{22}$$

Thus, the worst-case response time of message m is:

$$R_m = \max_{q=0..Q_m-1}(R_m(q)) \tag{23}$$

as in [2] (see (8)).

This analysis is safe but it may be pessimistic. Except for $q = 0$, we may be accounting the longest frame twice: Once explicitly in the last term of (22) and possibly a second time implicitly in $g_m(q)$ in (21).

We can remove this pessimism by modifying (22) to:

$$R_m(q) = J_m + w_m(q) - q\,T_m + g_m(q+1) - g_m(q) \tag{24}$$

That is, to compute the interference by higher-priority messages we assume that the first q message instances take the longest to transmit, and therefore the $q + 1$ instance will suffer maximum interference. On the other hand, by using $g_m(q+1) - g_m(q)$ in (24), we prevent the analysis from being pessimistic w.r.t. the cumulative transmission time of instances of message m.

V. Tighter Analysis

Note that the analysis in the previous section, even after the last change, although safe, may still be pessimistic. This is because we are considering the worst case for each instance q, but some of these worst cases may not occur simultaneously. For example, consider the multisized message example in Sec. IV-A. The worst case for $q = 0$ corresponds to the case where message subtype 1 is the first instance. However, for $q = 1$, the worst case happens when message subtype 0 is the first instance. It may be the case that when message subtype

0 is the first, the level-m busy-period is shorter than $T_m - J_m$ and therefore in that case, the sequence of message subtypes $0, 1$ will never occur in a level-m busy period.

We can avoid some of this pessimism, by computing the length of the level-m busy period for each of the S subtypes of message m as the first message instance in that busy period, and, for each of these busy periods, by computing the response time for each message instance in the busy period under consideration. With this approach, rather than solving just one recurrence per message m to compute the waiting time, we have to solve S recurrences. Thus, we can trade-off pessimism for analysis run time.

A. Analysis

In order to derive this tighter analysis we define $g_m(i,k)$, the cumulative transmission time of k consecutive instances of message m starting with subtype i, as:

$$g_m(i,k) = \sum_{j=i}^{i+k-1} C_m(j \bmod S) \tag{25}$$

where $C_m(j \bmod S)$ is the transmission time of subtype $j \bmod S$ of message m.

Likewise, we define $G_m(i,t)$, the cumulative worst-case transmission request of consecutive instances of message m starting with subtype i in a time interval with length t:

$$G_m(i,t) = g_m\left(i, \left\lceil \frac{t}{T_m} \right\rceil\right) \tag{26}$$

We can now compute the level-m busy period starting with subtype i of message m using the following recurrence:

$$t_m^{n+1}(i) = B_m + G_m(i, t_m^n(i) + J_m) + \sum_{\forall k \in hp(m)} G_k(t_m^n(i) + J_k) \tag{27}$$

A possible starting value is $t_m^0(i) = B_m + g_m(i,1)$. The number of instances of message m in $t_m(i)$ is given by:

$$Q_m(i) = \left\lceil \frac{t_m(i) + J_m}{T_m} \right\rceil \tag{28}$$

Again, for each instance q, ranging from 0 to $Q_m(i) - 1$, of message m in the level-m busy period we can compute its transmission start waiting time, using recurrence:

$$w_m^{n+1}(i,q) = B_m + g_m(i,q) + \sum_{\forall k \in hp(m)} G_k(w_m^n(i,q) + J_k + \tau_{bit}) \tag{29}$$

Possible starting values are $w_m^0(i,0) = B_m + \sum_{k \in hp(m)} g_k(1)$, and $w_m^0(i,q) = w_m(i,q-1) + g_m(i,q) - g_m(i,q-1)$, for $q \neq 0$.

The worst-case response time of instance q of a message sequence starting with subtype i of message m is given by:

$$R_m(i,q) = J_m + w_m(i,q) - q\,T_m + g_m(i,q+1) - g_m(i,q) \tag{30}$$

Thus, the worst-case response time, of an instance of m in a level-m busy period starting with subtype i of message m is given by:

$$R_m(i) = \max_{q=0..Q_m(i)-1} (R_m(i,q)) \tag{31}$$

Finally, the worst case response time of any instance of m in a level-m busy period is given by:

$$R_m = \max_{i=0..S-1} (R_m(i)) \tag{32}$$

VI. Examples

With the help of two example message sets, we demonstrate the process of response time computation and the differences between our proposed approach and the-state-of-the-art.

A. Example 1

This example illustrates our simpler analysis (Section IV) and compares it with the state-of-the art analysis.

Table II presents the attributes of the different messages in this example. The transmission times of the messages were selected among the values possible from Equation 9; the latter range from 55 (for a 0-byte message) to 135 (for an 8-byte message), with a step of 10. All message jitters are zero.

message m	priority	$T_m = D_m$	$\overrightarrow{C_m}$	C_m
message 1	high	200 τ_{bit}	{75,95,65}	95
message 2	medium	350 τ_{bit}	{55,75 }	75
message 3	low	400 τ_{bit}	{105,55}	105

TABLE II: A set of multisized CAN messages, used in our first example. Column $\vec{C}_m$ contains the vector of the transmission times of the different message subtypes, for use by the multisized CAN analysis. Column C_m contains the worst-case transmission time of all message instances (a scalar), for use by the classic CAN analysis. Message parameters are measured in multiples of τ_{bit}.

The worst case response time of message 2 is computed as follows. The blocking term for message 2 is $B_2 = 105$, which is the largest transmission time for all subtypes of message 3, the only message with lower priority. The busy period window t_2, computed with recurrence (3) (for the standard analysis) and recurrence (16) (for the multisized message CAN analysis), respectively converges to 540, (33) and 350, (34).

$$\begin{aligned} t_2^5 &= B_2 + \left\lceil \frac{540}{T_2} \right\rceil * C_2 + \left\lceil \frac{540}{T_1} \right\rceil * C_1 \\ &= 105 + \left\lceil \frac{540}{350} \right\rceil * 75 + \left\lceil \frac{540}{200} \right\rceil * 95 \\ &= 105 + 2 * 75 + 3 * 95 = 540 \end{aligned} \tag{33}$$

$$\begin{aligned} t_2^3 = &\quad B_2 + G_2(350) + G_1(350) \\ = &\quad 105 + 75 + (75 + 95) = 350 \end{aligned} \tag{34}$$

In this particular case, the number of instances of message 2 that need to be analyzed (i.e., that are contained in its busy period) is 2 for the standard analysis, see (5), and 1 for the multisized CAN message analysis, see (20).

The computation of the response time of the first instance of message 2 requires its queueing delay $w_2(0)$ to be computed first, using recurrences (6) for the standard analysis and (21) for our simpler analysis, which converge to 295 (35) and 275 (36), respectively:

$$\begin{aligned} w_2^3(0) &= B_2 + 0 * C_2 + \left\lceil \frac{295+1}{T_1} \right\rceil C_1 \\ &= 105 + 2 * 95 = 295 \end{aligned} \quad (35)$$

$$\begin{aligned} w_2^3(0) &= B_2 + g_2(0) + G_1(295) \\ &= 105 + 0 + (75 + 95) = 275 \end{aligned} \quad (36)$$

Applying the response time equations, respectively (7) and (24), we obtain 370 ($> D_2$) and 350 ($= D_2$). Thus, message 2 is not schedulable under the standard analysis, whereas the single instance of message 2 is schedulable under the multisized CAN message analysis of Section IV.

In summary, the worst-case response time of message 2 using the analysis of Section IV is 350 ($\leq D_2$) whereas using the classic analysis it is at least 370 ($> D_2$). This illustrates that, although the classic, state-of-the-art analysis provides safe upper bounds for message transmission times and response time, it can be pessimistic. As a result a system that is actually schedulable might be declared unschedulable, as demonstrated with this example.

B. Example 2

The differences between the simpler analysis of Section IV and the tighter, but more complex analysis of Section V can be more easily illustrated via a second example, with just two messages, one of which is multisized. Table III presents the attributes of those messages. All message jitters are zero. Recall also that $\tau_{bit} = 1$. Starting with our simpler analysis:

msg m	priority	T_m	D_m	$\overrightarrow{C_m}$	C_m
msg A	higher	160	235	{95}	95
msg B	lower	240	240	{65, 135,55}	135

TABLE III: The messages used in Example 2.

Message A has higher-priority, therefore it incurs no interference from message B. However, it has a blocking term B_A=$g_B(1)$=135 and it can suffer inteference from previously released but yet undelivered instances of itself (given that D_A>T_A). The busy period t_A at this priority level is the solution to the recurrence $t_A^{n+1} = B_A + \lceil \frac{t_A^n}{T_A} \rceil C_A$, i.e., 420.

There exist $Q_A = \lceil t_a/T_A \rceil = \lceil 420/160 \rceil = 3$ instances of message A in this busy period of 420, and applying Equation 22, their response times are respectively $135+95 = 230$, $135+2*95-1*160 = 165$, and $135+3*95-2*160 = 100$. Therefore the message deadline $D_A = 235$ is met. Onwards with the analysis of the lower-priority message B:

This message has the lowest priority, so its blocking term B_B is zero. We will first analyse its schedulability using the simpler analysis of Section IV. From (16), the busy period at the priority level of message B is given by the solution to the recurrence $t_B^{n+1} = G_B(t_B^n) + G_A(t_B^n)$, initiated by $t_B^0 = g_B(1) = 135$:

$$\begin{aligned} t_B^1 &= G_B(135) + G_A(135) = 135 + 95 = 230 \\ t_B^2 &= G_B(230) + G_A(230) = 135 + 2 \cdot 95 = 325 \\ t_B^3 &= G_B(325) + G_A(325) = (135 + 65) + 3 \cdot 95 = 485 \\ t_B^4 &= G_B(485) + G_A(485) = (135 + 65 + 55) + 4 \cdot 95 = 635 \\ t_B^5 &= G_B(635) + G_A(635) = (135 + 65 + 55) + 4 \cdot 95 = \mathbf{635} \end{aligned}$$

Therefore $t_B = 635$ and, from (20), there exist $Q_B = \lceil 635/240 \rceil = 3$ instances of message B in that busy period. Let us now find the time instants ($w_B(0)$, $w_B(1)$ and $w_B(2)$) when those message instances begin their transmission by applying and solving (21), and from those, their response times, via (24):

$$\begin{aligned} w_B^0(0) &= 0 \\ w_B^1(0) &= g_B(0) + G_A(0 + 0 + 1) = 0 + 95 = 95 \\ w_B^2(0) &= g_B(0) + G_A(95 + 0 + 1) = 0 + 95 = 95 \\ \Rightarrow R_B(0) &= w_B(0) - 0 * T_B + g_B(1) = 95 + 135 = 230 \end{aligned}$$

$$\begin{aligned} w_B^0(1) &= w_B^0(0) + g_B(1) - g_B(0) = 95 + 135 - 0 = 230 \\ w_B^1(1) &= g_B(1) + G_A(230 + 0 + 1) = 135 + 2 \cdot 95 = 325 \\ w_B^2(1) &= g_B(1) + G_A(325 + 0 + 1) = 135 + 3 \cdot 95 = 420 \\ w_B^3(1) &= g_B(1) + G_A(420 + 0 + 1) = 135 + 3 \cdot 95 = 420 \\ \Rightarrow R_B(1) &= w_B(1) - 1 * T_B + g_B(2) - g_B(1) \\ &= 420 - 240 + 200 - 135 = 245 \end{aligned}$$

Since $R_B(1) > D_B = 240$, message B is deemed unschedulable by the simpler analysis from Section IV. Let us now try the tighter analysis from Section V. Note that because message A is not multisized, there is no difference between the two analyses. So, we consider only message B. From (27), the recurrence for the busy period at message B's priority level, for a message sequence starting with subtype i, is:

$$t_B^{n+1}(i) = G_B(i, t_B^n(i)) + G_A(t_B^n(i))$$

When the subtype of the sequence-initial message is $i = 0$, starting with $t_B^0(0) = g_B(0, 1) = 65$:

$$\begin{aligned} t_B^1(0) &= G_B(0, 65) + G_A(65) = 65 + 95 = 160 \\ t_B^2(0) &= G_B(0, 160) + G_A(160) = 65 + 95 = \mathbf{160} \end{aligned}$$

Then $Q_B(0) = \lceil t_B(0)/T_0 \rceil = \lceil 160/240 \rceil = 1$, so there is a single instance of message B in this busy period – and since $t_B(0) \leq D_B$, it is schedulable.

When the subtype of the sequence-initial message is $i = 1$, starting with $t_B^0(1) = g_B(1,1) = 135$:

$$t_B^1(1) = G_B(1,135) + G_A(135) = 135 + 95 = 230$$
$$t_B^2(1) = G_B(1,230) + G_A(230) = 135 + 2 \cdot 95 = 325$$
$$t_B^3(1) = G_B(1,325) + G_A(325) = (135 + 55) + 3 \cdot 95 = 465$$
$$t_B^4(1) = G_B(1,465) + G_A(465) = (135 + 55) + 3 \cdot 95 = \mathbf{465}$$

Then, $Q_B(1) = \lceil t_B(1)/T_1 \rceil = \lceil 465/240 \rceil = 2$ instances of message B are in this busy period. From (29), they start their transmissions at instants $w_B(1,0)$ and $w_B(1,1)$ computed below, allowing their response times to be derived using (30):

$$w_B^0(1,0) = G_A(0 + 0 + 1) = 95$$
$$w_B^1(1,0) = g_B(1,0) + G_A(95 + 0 + 1) = 0 + 95 = \mathbf{95}$$

$$\Rightarrow R_B(1,0) = w_B(1,0) - 0 \cdot T_B + g_B(1,1) - g_B(1,0)$$
$$= 95 + 135 - 0 = 230$$

$$w_B^0(1,1) = w_B^0(1,0) + g_B(1,1) - g_B(1,0)$$
$$= 95 + 135 - 0 = 230$$
$$w_B^1(1,1) = g_B(1,1) + G_A(230 + 0 + 1) = 135 + 2 \cdot 95 = 325$$
$$w_B^2(1,1) = g_B(1,1) + G_A(325 + 1) = 135 + 3 \cdot 95 = 420$$
$$w_B^3(1,1) = g_B(1,1) + G_A(420 + 0 + 1) = 135 + 3 \cdot 95 = \mathbf{420}$$

$$\Rightarrow R_B(1,1) = w_B(1,1) - 1 * T + g_B(1,2) - g_B(1,1) =$$
$$= 420 - 240 + 190 - 135 = 235$$

Therefore, both instances of message B in a busy period initiated by subtype $i = 1$ are schedulable.

Similarly reasoning, the busy period for a sequence initiated by a subtype-2 message is $t_B(2) = 150$. It contains a single instance of message B and its response time is equal to the above busy period $t_B(2)$, so it is also schedulable. Therefore, $R_B = \max(R_B(0,0), R_B(1,0), R_B(1,1), R_B(2,0)) = \max(160, 230, 235, 150) = 235 < D_B$. Hence, the system is schedulable, using the tighter analysis.

VII. Conclusions

In this work, we showed how (i) expressing in a more fine-grained manner the transmission request requirements of CAN messages, without any modification to the CAN protocol and (ii) devising a schedulability analysis that is able to leverage the additional information can provide less pessimistic, but always safe, estimates on the worst-case response times of the CAN messages. This can make the difference between a system erroneously being deemed unschedulable or being proven schedulable. Alternatively, it can help safely accommodate additional functionality (and associated message traffic) in an existing system, by improving the safely attainable utilization. The multisized CAN message model and its analysis were motivated by practical considerations, as there can be cases of CAN messages whose size varies according to known cyclic patterns (e.g., to piggyback multiple signals with different periods). Our work drew from the multiframe task model by Mok and Chen and its analysis and transplanted those elements into the state-of-the-art schedulability analysis for the CAN bus.

Acknowledgements

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within CISTER Research Unit (UIDP/UIDB/04234/2020); by FCT and the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement through the European Regional Development Fund (ERDF), within project PREFECT (POCI-01-0145-FEDER-029119); by FCT through the European Social Fund (ESF) and the Regional Operational Programme (ROP) Norte 2020, under grant 2020.08045.BD.

References

[1] R. I. Davis, I. Bate, G. Bernat, I. Broster, A. Burns, A. Colin, S. Hutchesson, and N. Tracey, "Transferring real-time systems research into industrial practice: Four impact case studies," in *Proc. 30th Euromicro Conf. on Real-Time Systems (ECRTS)*, 2018, pp. 7:1–7:24.

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

[3] K. W. Tindell and A. Burns, "Guaranteeing message latencies on Controller Area Network (CAN)," in *Proc. 1st int. CAN conference*, 1994, pp. 1–11.

[4] K. Tindell, H. Hansson, and A. J. Wellings, "Analysing Real-Time Communications: Controller Area Network (CAN)," in *Proc. 15th IEEE Real-Time Systems Symposium (RTSS)*, 1994, pp. 259–263.

[5] K. W. Tindell, A. Burns, and A. J. Wellings, "Calculating Controller area network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163—-1169, 1995.

[6] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis, "Controller Area Network (CAN): Response time analysis with offsets," in *Proc. 9th WFCS*, 2012, pp. 43–52.

[7] R. I. 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 WFCS*, 2012, pp. 33–42.

[8] R. Sato and S. Fukumoto, "Response-time analysis for controller area networks with randomly occurring messages," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3893–3902, 2020.

[9] D. A. Khan, R. I. Davis, and N. Navet, "Schedulability analysis of CAN with non-abortable transmission requests," in *Proc. 16th IEEE ETFA*, 2011, pp. 1–8.

[10] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Schedulability analysis for Controller Area Network (CAN) with FIFO queues priority queues and gateways," *Real-Time Systems*, vol. 49, no. 1, pp. 73–116, 2013.

[11] A. K. Mok and D. Chen, "A multiframe model for real-time tasks," *IEEE Transactions on Software Engineering*, vol. 23, no. 10, pp. 635–645, 1997.

[12] S. K. Baruah, , and A. Mok, "Static-priority scheduling of multiframe tasks," in *Proc. 11th ECRTS*, 1999, pp. 38–45.

[13] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proc. 11th IEEE RTSS*, 1990, pp. 201–209.