Schedulability Analysis of CAN with Non-abortable Transmission Requests

Dawood A. KHAN
INRIA / INPL
Vandoeuvre, France
dawood.khan@loria.fr

Robert I. Davis
Department of Computer Science
University of York
York, UK
rob.davis@cs.york.ac.uk

Nicolas NAVET
INRIA / RealTime-at-Work
Vandoeuvre, France
nicolas.navet@inria.fr

Abstract

The analysis of the real-time properties of an embedded communication system relies on finding upper bounds on the Worst-Case Response Time (WCRT) of the messages that are exchanged among the nodes on the network. The classical WCRT analysis of Controller Area Network (CAN) implicitly assumes that at any given time, each node is able to enter its highest priority ready message into arbitration. However, in reality, CAN controllers may have some characteristics, such as non-abortable transmit buffers, which may break this assumption. This paper provides analysis for networks that contain nodes with non-abortable transmit buffers, as well as nodes that meet the requirements of the classical analysis. The impact on message WCRTs due to a limited number of transmission buffers with non-abortable behaviour is examined via two case-studies.

I. Introduction

Controller Area Network (CAN) was specifically designed for use in the automotive domain and has become a de-facto standard. Today, high-end cars can contain as many as 70 CAN controllers [1]. CAN has been extensively used in other areas as well, including industrial automation, especially networked control systems [2], because of its interesting real-time properties and low-cost. Whatever the domain, existing schedulability analyses of real-time applications distributed over CAN assume that:

1. If a CAN node has to send out a stream of messages having the highest priority on the bus, it should be able to do so without releasing the bus between two consecutive messages, despite the arbitration process that takes place at the end of each transmission.

2. If on a CAN node more than one message is ready to be sent, the highest priority message will be sent first. This means that the internal organization and message arbitration of the CAN node is such that this is possible. These assumptions put some constraints on the architecture of the CAN controllers and on the whole protocol stack. Sometimes, because of the CAN controller or protocol

layers, priority inversion among messages can occur. This can happen when the controller sends more distinct messages than the number of transmission buffers available and transmission requests (for low-priority messages) cannot be cancelled. Indeed, some CAN controller hardware implementations have internal organization such that they send messages independent of CAN message ID (Microchip MCP2515, Freescale MC68HC912), send messages in a FIFO order (Infineon XC161CS), or do not have enough transmit buffers (Philips SJA1000). Moreover, the transmit buffers may be managed without abortion (Philips 82C200) [3], or the support for abort mechanisms may be missing at the device driver level or, finally, the communication stack may be configured such that it does not support cancelling transmission (see "transmit cancellation" in an AUTOSAR stack, page 37 in [4]). As a result, a message can be delayed for a longer time than is expected by classical analyses [5], [6] and the response time increases accordingly.

II. Previous Work

Timing analyses of CAN developed over the years model the network as an infinite priority queue where each node is inserting its messages according to their priority. It is then assumed that the highest priority message in the queue wins the arbitration, be it in the deterministic [5], [6], [7] or stochastic case [8], [9]. However, this model does not hold when hardware and software constraints, like limited numbers of transmission buffers in the CAN controller and copy-time of messages from device drivers, are considered then the Worst-Case Response Time (WCRT) increases as compared to the traditional analyses. To the best of our knowledge, this issue was first identified and analysed in [10].

Some work has already been carried out to identify and analyse the effects of limited transmission buffers, in [10], [3], [11] and [12]. In [11], Natale classifies and explains all the cases leading to priority inversion due to hardware and software limitations, that were not covered by the existing analyses. In [10] Meschi et al. show that at least three transmission buffers are needed to avoid priority inversions when the copy-time of a message from the queue to the controller is neglected. However, analysis in [10] only addresses the case when transmission requests are abortable. In [12], Khan et al. address the case of priority inversion in an abortable CAN controller when copy-time of messages and the architecture of a device driver is taken into account. In [13], Davis et al. provide schedulability analysis when device drivers use FIFO transmission queues. However, the analyses provided in [12], [13] do not investigate the non-abortable CAN controller case. In [3] Natale provides an analysis for integrating the increase in WCRT due to priority inversion in non-abortable CAN controllers. However, the analysis provided in [3] takes into account interference from all lower priority messages when computing the WCRT of the message which suffers from priority inversion, which may not be the case as is shown in this paper. Furthermore, it does not consider the fact that the increase in the

WCRT (additional delay) of a message manifests itself as a jitter for lower priority messages.

Contributions of the paper: Here, we address the 3 or more buffer case when it is impossible to cancel a transmission request and we derive a worst-case response time analysis for it. The case addressed here is meaningful because in practice most CAN controllers have more than three buffers and the ability to cancel a transmission request may not be supported by them, the device drivers or the higher level communication stack. This work provides tighter bounds on the WCRT than derived in [3] by identifying more precisely the interference brought about by lower priority messages. It also identifies and integrates the jitter due to this interference in the analysis, which may increase the response time for some messages.

III. System model

We assume a set $\mathcal{M}$ of m messages μ_1, μ_2, ..., μ_m, where $m \in \mathbb{N}$. Each message μ_i is characterized by a *period* $T_i \in \mathbb{R}^+$, an activation jitter $J_i \in \mathbb{R}^+$, a worst-case transmission time $C_i \in \mathbb{R}^+$, and a (*relative*) *deadline* $D_i \in \mathbb{R}^+$, where $D_i \leq T_i$. For notational convenience, we assume that the messages are given in order of decreasing priority, i.e. μ_1 has highest priority and μ_m has the lowest priority. Moreover, we assume a set $\mathcal{C}$ of n CAN controllers CC_1, CC_2, ..., CC_n, where $n \in \mathbb{N}$. Each CAN controller CC_c has a finite number of transmission buffers $k_c \in \mathbb{N}$.

A total function $CC : \mathcal{M} \rightarrow \mathcal{C}$ defines which message is sent by which CAN controller. The set of messages M_c sent by controller CC_c is defined as

$$M_c = \{\mu \in \mathcal{M} | CC(\mu) = CC_c\}. \tag{1}$$

Similarly, $\overline{M_c}$ defines the set of messages *not* sent by CC_c, i.e.

$$\overline{M_c} = \{\mu \in \mathcal{M} | CC(\mu) \neq CC_c\} = \mathcal{M} \setminus M_c. \tag{2}$$

Let H_c be the set of highest priority messages in M_c excluding the k_c lowest priority messages. Similarly, let HE_c be the set of highest priority messages in M_c excluding the $k_c - 1$ lowest priority messages. We use μ_{L_c} to denote the lowest priority message in message set HE_c, where L_c is its priority. Furthermore, we assume that multiple transmission buffers on CAN controllers are not occupied by messages with the same priority. The assumption is made that nodes can always fill empty buffers with ready messages in time for the next bus arbitration.

The WCRT R_i of a message is defined as the maximum possible time taken by a message to reach the destination CAN controller, starting from the time of an initiating event responded to by the sending task. A message μ_i is said to be schedulable if and only if its WCRT R_i is less than or equal to the message relative deadline D_i and the system is schedulable if and only if all of the messages are schedulable.

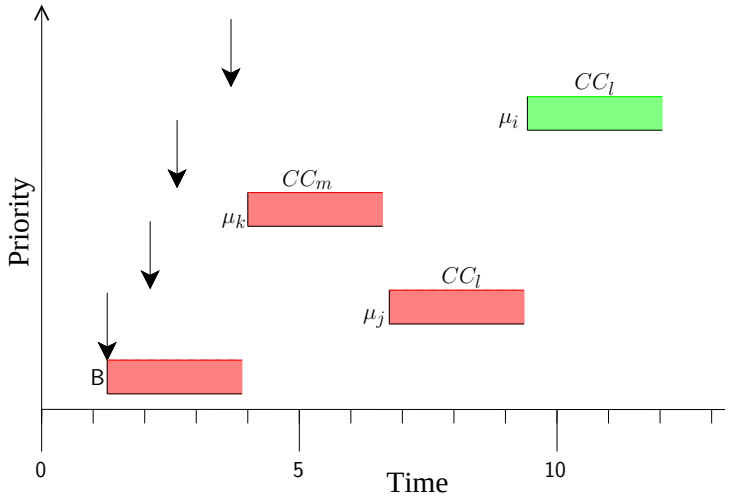

Figure 1. The message μ_i suffers a priority inversion as, being the highest priority message, it should have been transmitted earlier than μ_k and μ_j sent by nodes CC_m and CC_l respectively. This was not possible because here the transmission request for μ_j cannot be aborted on CC_l and all buffers were full. This results in an additional delay for message $\mu_i$and thus increased WCRT as compared to existing analyses. The arrows indicate the message release times.

Definition 1. [Priority inversion] A message μ_i on a CAN controller CC_l without abort mechanism is said to suffer from priority inversion when μ_i is released, if all of the k_l transmission buffers are occupied by the messages with lower priority than that of μ_i.

Remark 1. [Limited number of buffers] For any CAN controller CC_l with k_l transmission buffers the k_l lowest priority messages in the message set M_l will not suffer any priority inversion. As a corollary, for any CAN controller CC_l with k_l transmission buffers, if the number of messages mapped onto it is less than or equal to k_l then no message on CC_l can suffer from priority inversion.

IV. Additional Delay

Figure 1 illustrates the case in which a message μ_i sent by CAN controller CC_l should have been transmitted after B, the blocking time of a lower priority message. Here the message μ_j blocks μ_i due to the non-availability of a transmission buffer in CC_l, which only becomes available after μ_j finishes its transmission. However, the message μ_j has to wait for the higher priority message μ_k on CAN controller CC_m to be transmitted before it can begin its transmission. Therefore, the WCRT for μ_i given by the existing analyses increases by an amount, called the Additional Delay (AD), which in this example is equivalent to the sum of the worst-case transmission times of μ_k and μ_j.

Let μ_i be a high priority message in M_c and let the number of messages in M_c with a lower priority than i be at least k_c. Moreover, let μ_j be the highest priority message in the CC_c transmission buffers, such that $j > i$ (i.e. j is of lower priority than i). When all the transmission buffers of CC_c are full, the longest delay for μ_i occurs when none of the messages in the transmission buffers of CC_c are

currently being transmitted and μ_i has to wait until μ_j has been transmitted for the release of a buffer on CC_c. Moreover, μ_i also experiences the normal interference from higher priority messages sent by CAN controllers other than CC_c.

Algorithm 1 Algorithm for finding additional delay and additional jitter. The inputs to the algorithm are the number of CAN controllers (c), the number of transmission buffers on each CAN controller c (k_c), and the set of all messages on the CAN network (M). The algorithm returns the additional delay and additional jitter for all messages.

Input: c, $k = \{k_l | l = 1 \dots c\}$, M

Output: $AD = \{AD_i | i = 1 \dots size(M)\}, \hat{J} = \{\hat{J}_i | i = 1 \dots size(M)\}$

$AD = 0$ //initialization of AD for all messages

$\hat{J} = J$ //initialization of AJ for all messages

for *each* $CC_l | \, l \in \{1, 2 \dots, c\}$

$K = size(M_l)$ //$size(M_l)$ returns # of messages in M_l

$H_l = \{\forall \mu_i \in M | CC(\mu_i) == l \wedge i \leq K - k_l\}$ //set of messages with *AD*

if $K \leq k_l$ //more buffers available than the # of messages

$AD = 0$

else

$HE_l = \{\forall \mu_i \in M | CC(\mu_i) == l \wedge i \leq K - k_l + 1\}$ //message set H_l including μ_{L_l}

compute $R_j^* \forall \, \mu_j \in HE_l$ //using equations (3 & 5)

$\forall \, \mu_i \in < H_l$ *find* AD_i //using equation (6)

$\forall \, \mu_i \in < H_l$ *find* $\hat{J}_i = J_i + AJ_i$ //using equations (7 & 8)

end

end

return(AD *and* $\hat{J}$)

Before transmission (i.e. when μ_j is in the CAN controller transmission buffer blocking μ_i), μ_j can be directly blocked by at most one message μ_{l_j} with $l_j > j$ sent by another CAN controller, or alternatively, subject to indirect or push-through blocking due to at most one message μ_{l_j} with $l_j > j$ sent by the same CAN controller. Similarly, μ_j can experience interference from higher priority messages μ_{h_j} with $h_j < j$. Message μ_j cannot experience direct interference from higher priority messages μ_{h_j} with $h_j < j$ on controller CC_c, because μ_j is the highest priority message in the transmission buffers of CC_c and μ_j cannot be aborted. However, such messages could if transmitted prior to the time at which μ_j fills the buffer, cause indirect interference by delaying the transmission of higher priority messages sent by other nodes, which then increases the time taken for message

μ_j to be sent. To account for this indirect interference, we first include messages μ_{h_j} with $h_j < j$ on controller CC_c in the fixed point calculation of the queuing delay, so that the correct amount of interference is obtained for messages from other nodes. Later, when computing the additional jitter, we subtract out the interference from the messages sent by controller CC_c as these transmissions cannot occur after μ_j fills the transmission buffer.

The time duration for which μ_i has to wait depends on the response time of μ_j, called the modified response time[1] and denoted by R_j^* for μ_j and computed as follows

$$\hat{w}_j^{n+1} = \max(B_j, C_j) + \sum_{\forall \mu_k \in M \wedge k<j} \left\lceil \frac{\hat{J}_k + \hat{w}_j^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{3}$$

where B_j is the maximum blocking time of message μ_j given by:

$$B_j = \max\{0, \max\{C_k | k > i\}\}. \tag{4}$$

Where $\hat{J}_k$ is the jitter[2] of higher priority messages computed using equation (7) by algorithm 1. A suitable starting value for the recurrence relation given in equation (3) is $\hat{w}_j^0 = \bar{B}_j$. This relation keeps on iterating until $\hat{w}_j^{n+1} = \hat{w}_j^n$ or $\hat{w}_j^{n+1} + C_j > D_j$, which is the case when μ_j is not schedulable. The modified WCRT of μ_j is given by:

$$R_j^* = \hat{w}_j + C_j \tag{5}$$

There are some aspects that need to be taken into account in order to determine the additional delay experienced by μ_i, due to the non-availability of a transmission buffer. First, the jitter J_j of μ_j should not be accounted for in the modified WCRT R_j^* of μ_j, because that is irrelevant for the delay of μ_i as μ_j is already in the transmission buffer.

Second, because the interference of messages μ_{h_i} with $1 \leq h_i < i$ will re-appear when we compute the worst-case response time of μ_i, we have to *subtract* this interference from R_j^*, in order to prevent the double inclusion of interference from the messages μ_{h_i} with $1 \leq h_i < i$ sent by other CAN controllers (i.e. $\bar{M}_c$).

The *additional delay* AD_i of μ_i, due non-availability of transmission buffer, is therefore found by *subtracting* the interference of the messages μ_{h_i} with $1 \leq h_i < i$ and μ_{h_k} with $1 \leq h_k < j$ contained

[1]The modified response time of message μ_j is not its actual response time because the message jitter is missing.

[2]To begin with $\hat{J}_k = J_k$ for all messages, in order to find the first value of AD_i. After computing AD_i, it will appear as jitter to all messages $\{\mu_k | k > i\}$ necessitating recalculation of AD_i, which is done iteratively until it does not change any more or a message becomes unschedulable, found using algorithm 1.

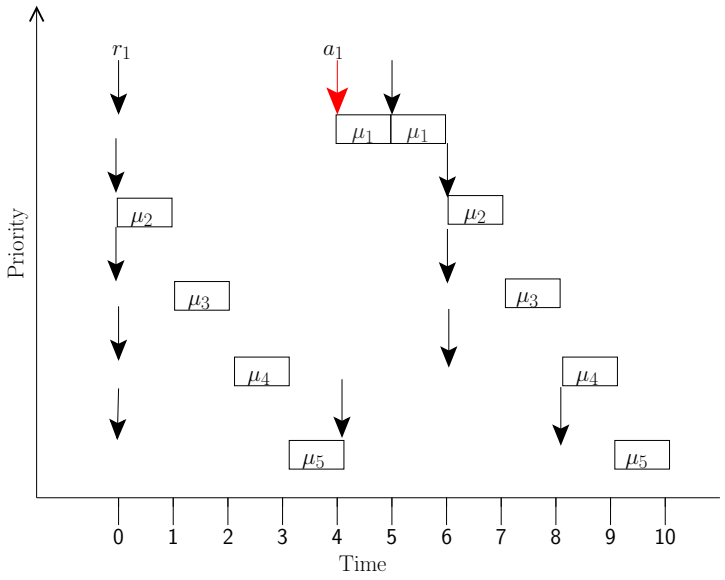

Figure 2. Example of how the WCRT of a lower priority message μ_5 is affected by the additional jitter caused by priority inversion that is suffered by a higher priority message μ_1.

in R_j^*, i.e.

$$AD_i = \max_{\forall k > i \wedge \mu_k \in HE_c} \left(R_k^* - \sum_{1 \le h_i < i \wedge \mu_{h_i} \in \bar{M}_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_i} + \tau_{\text{bit}}}{T_{h_i}} \right\rceil C_{h_i} - \sum_{1 \le h_k < k \wedge \mu_{h_k} \in M_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_k} + \tau_{\text{bit}}}{T_{h_k}} \right\rceil C_{h_k} \right) \quad (6)$$

The reason for taking max in equation (6) is that the additional delay for the message μ_i can be due to each message $\mu_k \in HE_c$ where $i < k \le L_c$, and it may be different due to each of these messages. Moreover, for all messages μ_k, such that $i < k \le L_c$, having similar higher priority interference to that of μ_{L_c} (i.e. $R_k^* - C_k$ is equal to $R_{L_c}^* - C_{L_c}$) the worst-case AD_i is obtained by taking into account the message μ_k with the largest worst-case transmission time (*i.e.* $C_k > C_{L_c}$), as μ_k will give more additional delay than μ_{L_c}. Thus taking the maximum over all messages which could block μ_i enables us to find the message μ_k with $i < k \le L_c$ which gives the worst-case additional delay to μ_i. The algorithm to find the additional delay is described in algorithm 1. The algorithm will keep on iterating until AD converges or it is greater than the deadline, *i.e.* WCRT of the message becomes greater than its deadline (in which case the message set is not schedulable).

Figure 3. The time line of message μ_i from its initiating event until it is able to participate in bus arbitration.

V. Additional Jitter

The release jitter (J_i) is defined traditionally as the time interval between the occurrence of an event that will trigger sending of the message (r_i) and placing the message in a transmission queue (Q) or a transmission buffer. However, with non-abortable transmit buffers, priority inversion occurs, and the message μ_i triggered by the event at r_i is not able to participate in arbitration until the time a_i, as it may be blocked by messages with lower priority than i. Therefore, the messages on other nodes see the interference of μ_i after time a_i and the jitter of this message is not limited to J_i. Instead, the total jitter seen for μ_i, by the messages with lower priority than i, is given by:

$$\hat{J}_i = J_i + AJ_i \tag{7}$$

where AJ_i is the time μ_i has to wait for the buffer to be emptied, see figure 3. Where AJ_i is computed as:

$$AJ_i = \max_{\forall k>i \wedge \mu_k \in HE_c} (R_k^* - \sum_{1 \le h_k < k \wedge \mu_{h_k} \in M_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_k} + \tau_{\text{bit}}}{T_{h_k}} \right\rceil C_{h_k}) \tag{8}$$

where R_k^* is found using equation (5). Note that interference from higher priority messages sent by the same node is subtracted out, as this interference cannot occur after message μ_k has filled the transmit buffer. The above equation upper bounds the amount of time that a message μ_k can spend in a transmit buffer, with all other buffers filled by lower priority messages; hence it upper bounds the additional delay caused by message μ_k on message μ_i .

Table I

Characteristics of messages.

Frames	CAN controller	T	J	C
μ_1	CC_1	$5C$	0	C
μ_2	CC_2	$6C$	0	C
μ_3	CC_2	$6C$	0	C
μ_4	CC_2	$6C$	0	C
μ_5	CC_1	$4C$	0	C

Example 1. Consider a system of two CAN controllers CC_1 and CC_2 with 5 messages, as described in table I. Let CC_1 have a single transmission buffer and let CC_2 have an unlimited number of transmission buffers. Assume that μ_5 is in the buffer of CC_1 and μ_1 is released along with all other messages at time $t = 0$, see figure 2. Since CC_1 has a single buffer, μ_1 is blocked until μ_5 releases the buffer at time $t = 4$. The messages with lower priority than that of μ_1 on CC_2 are not aware of release at $t = 0$ of μ_1, as they do not see it participating in arbitration from $t = 0$ to a_1 when it occupies the buffer in CC_1. Once μ_1 is in the buffer it is able to participate in arbitration at time $t = 4$ and wins. The release of the second instance of message μ_5 suffers interference from two instances of message μ_1, between time $t = 4$ and $t = 6$. The inter-arrival time expected for μ_1 was $5C$, however, because μ_1 suffered an additional delay of $4C$ due to priority inversion, the interval between two instances of message μ_1 being sent on the bus is reduced to $1C$. The additional delay suffered by μ_1 is seen as a jitter of $4C$ by μ_5. The WCRT of μ_5 given by existing analyses is $5C$, but if we include the jitter of $4C$ for μ_1 we obtain the WCRT of $6C$ for μ_5 as seen in figure 2.

VI. Response time analysis

This section provides a method for computing the worst-case response time of messages on the CAN network. The computed values are then used to check the schedulability of the system by comparing the WCRTs against the message deadlines. The analysis given in this paper provides a simple and non-necessary schedulability condition directly inspired by [6]. It assumes no errors on the bus but they can be included as done in [5]. Following the analyses given in [5], [6] the worst-case response time can be described as a composition of three elements:

1) the queuing jitter J_i, is the maximum time between the sending task being released and a message being queued.
2) the queuing delay w_i, is the longest time for which a message can remain in the device driver queue or transmission buffers before successful transmission,
3) the worst-case transmission time C_i, is the longest time a message can take to be transmitted.

A bound on the worst-case response time of a message μ_i is therefore given by:

$$R_i = J_i + w_i + C_i \tag{9}$$

When computing a bound on the response time, we can distinguish three cases i) messages which are safe from priority inversion ii) messages which suffer from priority inversion due to non-abortion of the messages in transmission buffers and iii) messages which suffer from priority inversion due to copy-time and message swapping issues. In this paper, we cover case (i) and case (ii), while the third case has already been analysed in [12].

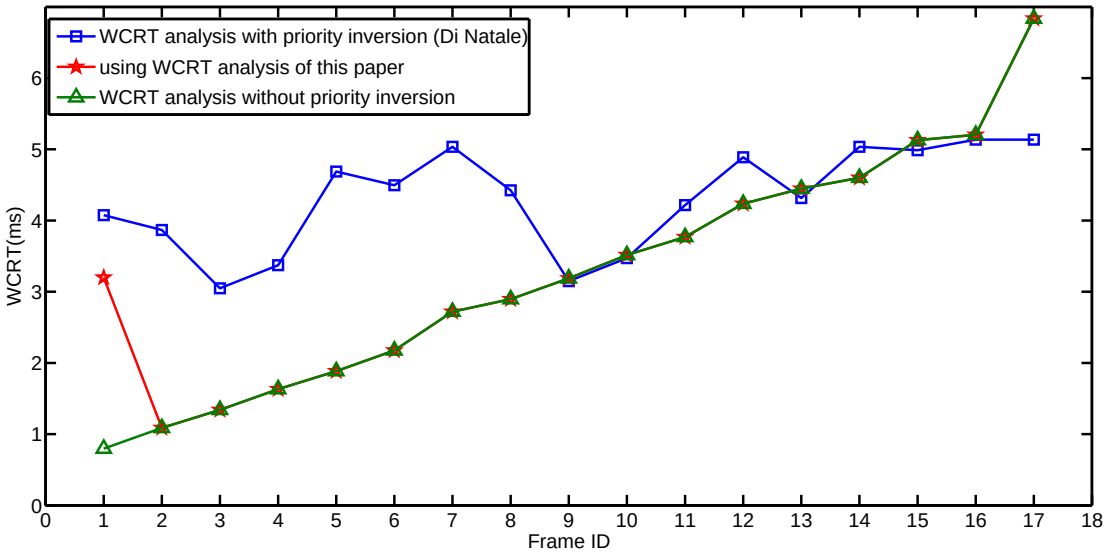

Figure 4. This figure shows the WCRT of messages from SAE benchmark computed using analysis which does not account for priority inversion, analysis in [3] and the analysis developed in this paper. Our analysis assumes each CAN controller has 3 transmission buffers. Some of the messages have lower WCRT with Di Natale's analysis (for example IDs 13, 15 and 17) because the equation used in [3] to compute the WCET is slightly different.

The queuing delay $w_i$is composed of:

1) blocking delay[3] $\hat{B}_i$, is either the delay B_i due to the non-preemptivity of lower priority messages in transmission when μ_i was ready for arbitration or the additional delay AD_i, computed using equation (6), due to the priority inversion i.e.

$$\hat{B}_i = \max(\max(B_i, C_i), AD_i) \tag{10}$$

2) the delay due to interference of higher priority messages which may win arbitration and be transmitted before μ_i.

A. *Case 1: safe from any priority inversion*

We note that the higher priority messages on each CAN controller CC_l are more susceptible to priority inversion than lower priority messages on the same CAN controller. Indeed, the k_l lowest priority messages on CC_l will not suffer from any priority inversion as not all of the transmission buffers can be occupied by messages with lower priority than any of these k_l messages, thus these messages are not suffering from any *additional delay*. However, these messages are still affected by

[3]The additional delay AD_i of a message μ_i appears as an additional blocking delay due to messages with lower priority than that of μ_i.

the additional delay of higher priority messages, as it is seen by them as additional jitter. For these messages or the CAN controllers which support abort mechanisms, the worst-case queuing delay, using the model in [6], is given by:

$$w_i^{n+1} = \max(B_i, C_i) + \sum_{\forall k<i \wedge \mu_k \in M} \left\lceil \frac{\hat{J}_k + w_i^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{11}$$

where $\hat{J}_k$ is computed using (7) and B_i is the maximum blocking time due to lower priority messages which occurs when a lower priority message of the largest size has just started to be transmitted when μ_i arrives, i.e.

$$B_i = \max_{\forall k>i \wedge \mu_k \in M} \{C_k\} \tag{12}$$

A suitable starting value for the recurrence relation given above is $w_i^0 = C_i$. This relation keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. If the message is schedulable its WCRT is given by (9).

We observe that the existing priority assignment algorithms, see [14], may not be optimal in this case as they require that the relative order among the higher priority messages does not matter while checking the schedulability of lower priority messages. However, such a condition is not satisfied, for the scenario discussed in this paper, as the order among the higher priority messages may impact their additional delay, i.e. the jitter $\hat{J}$ seen by lower priority messages, thus having an impact on the response time of lower priority messages.

B. Case 2: not safe from priority inversion

Once we have the additional delay of message μ_i, susceptible to priority inversion, we can compute its WCRT. The worst-case queuing delay for message μ_i is given by:

$$w_i^{n+1} = \hat{B}_i + \sum_{\forall k<i \wedge \mu_k \in M} \left\lceil \frac{\hat{J}_k + w_i^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{13}$$

where $\hat{J}_k$ is computed using (7) and $\hat{B}_i$ is computed using (10). A suitable starting value for the recurrence relation given above is $w_i^0 = C_i + AD_i$. This relation keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. If the message is schedulable its WCRT is given by (9).

However, as we established in section V the computed additional jitter for μ_i now impacts all the messages with lower priority than i and therefore we have to re-compute the WCRT[4] for all lower

[4]It is important to note that the additional delays effectively increase the jitter of affected messages, and this then leads to higher interference and a larger computed response time. However, in practice, the messages cannot obtain their maximum jitter (additional delays) all at the same time and therefore the analysis can be pessimistic. An improvement to the analysis is to upper bound the WCRT by the longest busy period at the lowest priority level, since no response time can be larger than that with any non-idling policy.

Algorithm 2 Algorithm for finding WCRT. The inputs to the algorithm are the number of CAN controllers (c), the number of transmission buffers on each CAN controller c (i.e. k_c), and the set of all messages on the CAN network (M). The algorithm returns the WCRT of message set.

Input: c, $k = \{k_l | l = 1 \dots c\}$, M

Output: WCRT of message set M

AD, $AD^{old} = 0$ // initialization of AD for all messages

$AD^{new} = C$

$\hat{J} = J$ // initialization of jitter for all messages

while(AD^{new}not equal to AD^{old})

$AD^{old} = AD^{new}$

Compute $\hat{J}$, AD^{new} via algorithm 1

if(AD^{new} is greater than deadlines)

return(unschedulable)

end

end

$AD = AD^{new}$

if($J + w^{n+1} + C \leq D$) //for case 1 and case 2 using equations (9, 11 & 13)

return($J + w^{n+1} + C$)

else

return(unschedulable)

end

priority messages as well.

The process used to re-compute WCRT for the messages remains the same as described in sections VI-A and VI-B. A simple procedure is used to find the WCRT by computing additional delay first (for all messages susceptible to priority inversion) and then computing the WCRT for all of the messages, as shown in algorithm 2.

Example 2. In section V we showed, with the aid of an example, how the additional delay of a message manifests itself as a jitter for lower priority messages and how existing analyses fail to integrate the same. We return to the same example to illustrate how the analysis developed in this paper integrates the additional delay and the additional jitter. The message μ_1 is blocked by μ_5 and therefore the additional delay for μ_1 calculated using equation (6) is $4C$. The WCRT for μ_1 computed by equation (13) is $5C$. Similarly, the WCRT of message μ_5 when computed using equation (11) (by accounting for the additional jitter of message μ_1) is $6C$, which can be verified from figure 2.

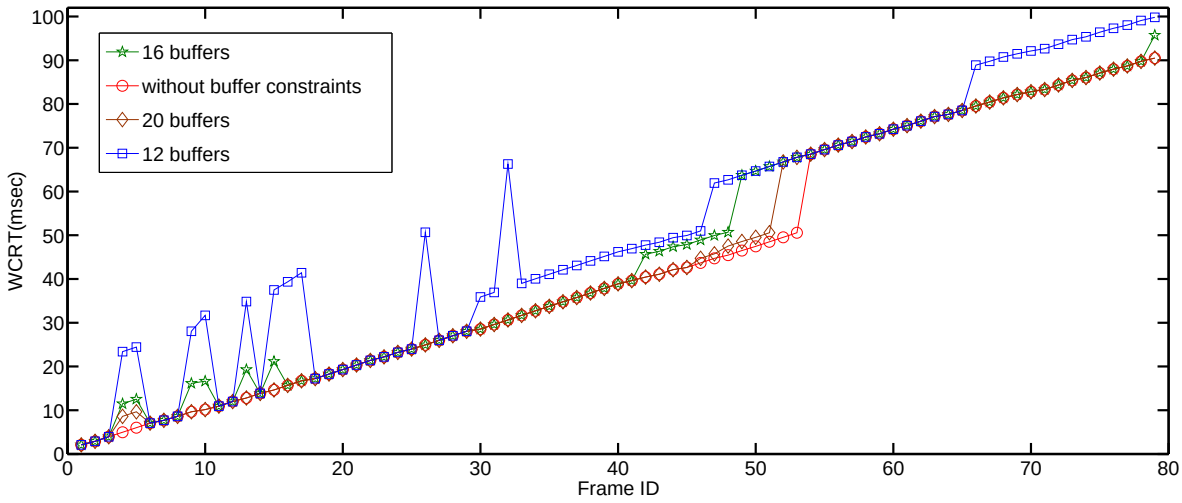

Figure 5. WCRT on a typical 125 kbits/s automotive body network (assuming each CAN controller has 12, 16 and 20 transmission buffers and cancellation of transmit request is not possible) computed using analysis which does not account for priority inversion (lower curve) and analysis developed in this paper.

VII. Comparative Evaluation

The analysis developed in this paper is compared against the existing analyses which do not account for priority inversion, and the analysis developed in [3] which accounts for priority inversion. The case-study in this paper assumes 3 or more transmission buffers on each CAN controller, with non-abortable transmission requests.

A. SAE benchmark

The evaluation of the analysis developed in this paper is done by comparing against SAE benchmark results published in [3] and in [15]. The SAE benchmark, see [15], [3] for details, describes a message set mapped on to seven different CAN controllers in a prototype car and the requirements for the schedulability of the messages. The network connecting the car subsystems handles 53 periodic and sporadic real-time signals. The signals have been grouped and the entire set has been reduced to 17 messages (for details, refer to [15]). To analyse the schedulability of the message set at $250\,kbps$ we compute the worst-case transmission time for this bus-speed, which for consistency is computed as in [3]. The results of the comparative WCRT analyses have been depicted in figure 4. The message set is schedulable with the analysis given in [3] and with the analysis provided in this paper. However, a significant difference in the response time computed by the analysis in this paper and the analysis in [3] can be observed in figure 4. The reason for such a difference is that the analysis in [3] does not consider the number of transmission buffers and computes the additional delay of the messages using

the lowest priority message from the message set mapped onto that CAN controller, thus resulting in a pessimistic WCRT. Moreover, it has been established in [12] and this paper that the number of transmission buffers does have an effect on the WCRT. Applying the criteria developed for priority inversion in this paper we find only one message in the benchmark may suffer from priority inversion ($ID = 1$), since there is only one CAN controller that has more than three messages mapped to it (see message mapping details in [3]). Thus, the WCRT only increases for the message with $ID = 1$ as the rest of the messages are safe from priority inversion and they only take into account the additional jitter of the message with $ID = 1$. The worst-case for message $ID = 1$ is when the transmission buffers are filled with messages of $ID = 8, 12, 15$. The first message to transmit from the buffers is then $ID = 8$, which contributes towards the worst-case additional delay for message $ID = 1$, as in the worst-case it may have to wait for higher priority messages from other CAN controllers to be transmitted first (i.e. $ID = 2, 3, 4, 5, 6, 7$ contribute additional delay, computed using equation (6)).

B. Automotive body network

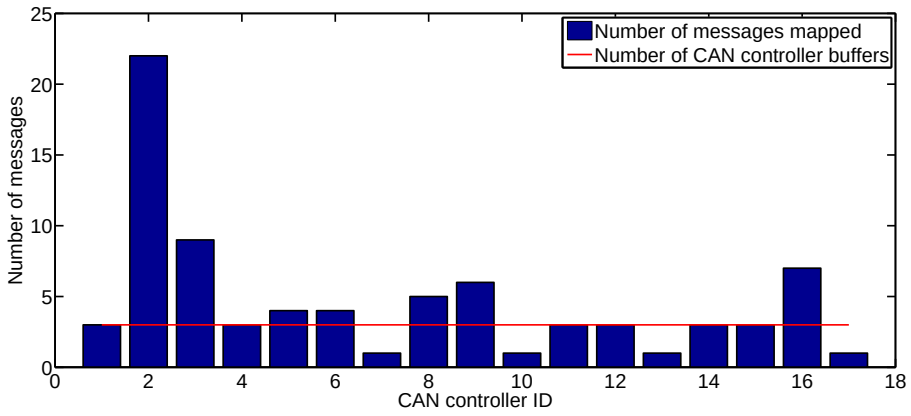

Figure 6. Figure showing number of messages mapped onto each CAN controller. The CAN controllers with more messages than the number of transmission buffers are susceptible to priority inversion.

The limitation of the SAE benchmark is that it is outdated with respect to current in-vehicle systems. Moreover, the SAE benchmark has only one node with more than 3 messages mapped onto it, thus making it difficult to compare the analyses. Therefore, we illustrate the new analysis on an typical 125Kbit/s automotive body network. To generate a realistic test configuration we used the *Netcarbench* [16] benchmark generator. The generated periodic message set under study consists of 79 CAN messages mapped over 17 ECUs with deadlines equal to periods and data payload ranging from 1 to 8 bytes. The total periodic load is equal to 64.26%. Figure 6 shows the message load distribution over the ECUs highlighting the ECUs with more than three messages susceptible to priority inversion, in the case where each node has three buffers. Figure 5 shows the worst-case response time of the

CAN messages with and without priority inversion. We observe the impact on the WCRT of messages when priority inversion is taken into account. For instance, the message set is unschedulable when 3 transmission buffers per node is considered. Moreover, in figure 5, the WCRT for the message with ID=32 when considering 12 transmission buffers raises from 30.64ms without priority inversion to 66.29ms. The underlying reason for such an increase in the WCRT is the additional delay of 19.46ms encountered by frame ID=32. This is because the frame which is blocking message ID=32 in the worst-case scenario has ID=69 and the number of frames on other ECUs having ID between ID=69 and ID=32 is 27. Therefore, in the worst-case additional delay scenario, 27 messages may be transmitted before message ID=69 could be transmitted and then subsequently release the buffer for message ID=32.

We also note that the choice of priorities greatly influences the amount of additional delay. For example, if the priorities were such that the message blocking the message with ID=32 in worst-case had ID=44, then the number of messages on other ECUs blocking message ID=32 would have been reduced to 10 from 27, resulting in a smaller additional delay.

VIII. Conclusion

The paper provides an analytical model of schedulability analysis for CAN controllers when the transmission requests cannot be aborted. The model developed in this paper provides understanding of the consequences of architectural limitations in CAN. Here, we derive a more realistic response time analysis in the typical case where controllers have three or more transmission buffers and do not possess the ability to cancel transmission requests. This analysis is of particular interest to automotive sector where multiple Tier 1 suppliers provide ready-to-use ECUs in an automobile. The lack of knowledge at system design level about the limitations of the CAN controller used or device driver provided by tier 1 suppliers can have serious consequences. A first follow-up to this paper is to come up with an analysis valid in the arbitrary deadline case. Another direct follow-up to this study is to investigate the case where, due to a larger message copy time, the nodes are not always able to fill empty buffers with ready messages in time for the next arbitration. The choice of the priorities has a direct effect on the additional delay due to requests that cannot be aborted. Therefore, in a future work, we aim to develop a priority assignment algorithm based on the schedulability test in this paper, which could reduce the amount of additional delay in the case where a message suffers from priority inversion. Also, we will study the choice of offsets on ECUs so that messages are not released at the same moment, reducing priority inversion in the CAN controller.

Acknowledgments

The authors would like to thank Reinder J. Bril (Eindhoven University of Technology) for the useful discussions on an earlier version of this paper.

REFERENCES

[1] N. Navet, Y.-Q. Song, F. Simonot Lion, and C. Wilwert, "Trends in Automotive Communication Systems," *Proceedings of the IEEE*, vol. 93, pp. 1204–1223, Jun 2005.

[2] P. Marti and, A. Camacho, M. Velasco, and M. El Mongi Ben Gaid, "Runtime Allocation of Optional Control Jobs to a Set of CAN-Based Networked Control Systems," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 503–520, November 2010.

[3] M. D. Natale, "Evaluating message transmission times in Controller Area Networks without buffer preemption," in *8th Brazilian Workshop on Real-Time Systems*, 2006.

[4] AUTOSAR, "Specification of CAN driver." Autosar Release 4.0 Rev1. Available at http://www.autosar.org, 2009.

[5] K. Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163 – 1169, 1995.

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

[7] M. Grenier and N. Navet, "Fine-tuning MAC-level protocols for optimized real-time QoS," *IEEE Transactions on Industrial Informatics*, vol. 4, pp. 6 –15, February 2008.

[8] H. Zeng, M. Di Natale, P. Giusto, and A. Sangiovanni-Vincentelli, "Using Statistical Methods to Compute the Probability Distribution of Message Response Time in Controller Area Network," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 678 –691, November 2010.

[9] H. Hansson, T. Nolte, C. Norstrom, and S. Punnekkat, "Integrating Reliability and Timing Analysis of CAN-Based Systems," *IEEE Transactions on Industrial Electronics*, vol. 49, pp. 1240–1250, Dec. 2002.

[10] A. Meschi, M. D. Natale, and M. Spuri, "Priority inversion at the network adapter when scheduling messages with earliest deadline techniques," in *8th Euromicro Workshop on Real-Time Systems*, pp. 243–248, June 1996.

[11] M. D. Natale, "Understanding and using the Controller Area Network." Handout of a lecture at U.C. Berkeley available at http://inst.eecs.berkeley.edu/~ee249/fa08/, October 2008.

[12] D. A. Khan, R. J. Bril, and N. Navet, "Integrating hardware limitations in CAN schedulability analysis," in *Wip paper at the 8th IEEE International Workshop on Factory Communication Systems (WFCS 2010)*, pp. 207–210, May 2010.

[13] R. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller Area Network (CAN) schedulability Analysis with FIFO queues," in *23rd Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 45–56, 5-8th July 2011.

[14] R. Davis and A. Burns, "Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," *Real-Time Systems*, vol. 47, pp. 1–40, 2011.

[15] K. Tindell and A. Burns, "Guaranteeing message latencies on Controller Area Network (CAN)," in *Proceedings of 1st international CAN conference*, pp. 1–11, 1994.

[16] C. Braun, L. Havet, and N. Navet, "NETCARBENCH: a benchmark for techniques and tools used in the design of automotive communication systems," in *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, pp. 321–328, 2007. Available at http://www.netcarbench.org.