Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2013, Article ID 148015, 11 pages
http://dx.doi.org/10.1155/2013/148015

Review Article

Response Time Analysis of Messages in Controller Area Network: A Review

Gerardine Immaculate Mary,[1] Z. C. Alex,[1] and Lawrence Jenkins[2]

[1] *School of Electronics Engineering, VIT University, Vellore, Tamilnadu 632014, India*
[2] *Department of Electrical Engineering, IISc, Bangalore, Karnataka 560012, India*

Correspondence should be addressed to Gerardine Immaculate Mary; gerardine@yahoo.com

Received 29 August 2012; Accepted 12 December 2012

Academic Editor: Zhiyong Xu

This paper reviews the research work done on the response time analysis of messages in controller area network (CAN) from the time CAN specification was submitted for standardization (1990) and became a standard (1993) up to the present (2012). Such research includes the worst-case response time analysis which is deterministic and probabilistic response time analysis which is stochastic. A detailed view on both types of analyses is presented here. In addition to these analyses, there has been research on statistical analysis of controller area network message response times.

1. Introduction

The arbitration mechanism employed by CAN means that messages are sent as if all the nodes on the network share a single global priority-based queue. In effect, messages are sent on the bus according to fixed priority nonpreemptive scheduling [1]. In the early 1990s, a common misconception was that although the protocol was very good at transmitting the highest priority messages with low latency, it was not possible to guarantee that the less urgent signals carried in lower priority messages would meet their deadlines [1]. In 1994, Tindell et al. [2–5] showed how research into fixed priority preemptive scheduling for single processor systems could be applied to the scheduling of messages on CAN. This analysis provided a method of calculating the worst-case response times of all CAN messages. Using this analysis it became possible to engineer CAN-based systems for timing correctness, providing guarantees that all messages and the signals that they carry would meet their deadlines. In 2007, Davis et al. [1] refuted this analysis and showed that multiple instances of CAN messages within a busy period (this period begins with a critical instant) need to be considered in order to guarantee that the message and the signals that they carry would meet their deadlines, since CAN effectively implements fixed priority nonpreemptive scheduling of messages.

Real-time researchers have extended schedulability analysis to a mature technique which for non-trivial systems can be used to determine whether a set of tasks executing on a single CPU or in a distributed system will meet their deadlines or not [1, 2, 4, 5]. The essence of this analysis is to investigate if deadlines are met in a worst-case scenario. Whether this worst case actually will occur during execution, or if it is likely to occur, is not normally considered [6].

In contrast with schedulability analysis, reliability modelling involves the study of fault models, the characterization of distribution functions of faults, and the development of methods and tools for composing these distributions and models in estimating an overall reliability figure for the system [6].

This separation of deterministic (0/1) schedulability analysis and stochastic reliability analysis is a natural simplification of the total analysis. This is because the deterministic schedulability analysis is quite pessimistic, since it assumes that a missed deadline in the worst case is equivalent to always missing the deadline, whereas the stochastic analysis extends the knowledge of the system by computing how often a deadline is violated [7].

There are many other sources of pessimism in the analysis, including considering worst-case execution times and worst-case phasings of executions, as well as the usage of pessimistic fault models. In a related work [8], a model for calculating worst-case latencies of controller area network (CAN) frames (messages) under error assumptions is proposed. This model is pessimistic, in the sense that there are systems that the analysis determines to be unschedulable, even though deadlines will be missed only in extremely rare situations with pathological combinations of errors.

In [9, 10] the level of pessimism is reduced by introducing a better fault model, and in [9] variable phasings between message queuing are also considered, in order to make the model more realistic. In [11] the pessimism introduced by the worst-case analysis of CAN message response times is reduced by using bit-stuffing distributions in the place of the traditional worst-case frame sizes which are referred to in [6, 7].

The organization of the paper is as follows: in Section 2, the review of the research on Worst Case Response Time Analysis of CAN messages is presented, and in Section 3, the review of the research on Probabilistic Response Time Analysis of CAN messages is presented. In both sections, the method of bit stuffing is reviewed.

2. Worst-Case Response Time Analysis of CAN Messages

In automotive applications, the *messages* sent on CAN are used to communicate state information, referred to as *signals*, between different ECUs. Examples of signals include wheel speeds, oil and water temperature, engine rpm, gear selection, accelerator position, dashboard switch positions, climate control settings, window switch positions, fault codes, and diagnostic information. In a high-end vehicle there can be more than 2500 distinct signals, each effectively replacing what would have been a separate wire in a traditional point-to-point wiring loom.

Many of these signals have real-time constraints associated with them. For example, an ECU reads the position of a switch attached to the brake pedal. This ECU must send a signal, carrying information that the brakes have been applied, over the CAN network so that the ECU responsible for the rear light clusters can recognise the change in the value of the signal and switch the brake lights on. All this must happen within a few tens of milliseconds of the brake pedal being pressed. Engine, transmission, and stability control systems typically place even tighter time constraints on signals, which may need to be sent as frequently as once every 5 milliseconds to meet their time constraints [1]. Hence it is essential that CAN messages meet their deadlines.

2.1. Related Work. CAN is a serial data bus that supports priority-based message arbitration and non-pre-emptive message transmission. The schedulability analysis for CAN builds on previous research into fixed priority scheduling of tasks on single processor systems [12].

In 1990, Lehoczky [13] introduced the concept of a busy period and showed that if tasks have deadlines greater than their periods (referred to as *arbitrary deadlines*) then it is necessary to examine the response times of all invocations of a task falling within a busy period in order to determine the worst-case response time. In 1991, Harbour et al. [14] showed that if deadlines are less than or equal to periods, but priorities vary during execution, then again multiple invocations must be inspected to determine the worst-case response time. We note that non-pre-emptive scheduling is effectively a special case of pre-emptive scheduling with varying execution priority—as soon as a task starts to execute, its priority is raised to the highest level. In 1994, Tindell et al. [12] improved upon the work of Lehoczky [13], providing a formulation for arbitrary deadline analysis based on a recurrence relation.

Building upon these earlier results, comprehensive schedulability analysis of non-pre-emptive fixed priority scheduling for single processor systems was given by George et al. in 1996 [15]. In 2006, Bril [16] refuted the analysis of fixed priority systems with deferred pre-emption given by Burns in [17], showing that this analysis may result in computed worst-case response times that are optimistic. The schedulability analysis for CAN given by Tindell et al. in [2–5] builds upon [17] and suffers from essentially the same flaw. A similar issue with work on pre-emption thresholds [18] was first identified and corrected by Regehr [19] in 2002. A technical report [20] and a workshop paper [21] highlight the problem for CAN but do not provide a specific in-depth solution.

The revised schedulability analysis presented in [1] aims to provide an evolutionary improvement upon the analysis of CAN given by Tindell et al. in [2–5]. To do so, it draws upon the analysis of Tindell et al. [12] for fixed priority pre-emptive scheduling of systems with arbitrary deadlines, and the analysis of George et al. [15] for fixed priority non-pre-emptive systems, and also presents a sufficient but not necessary schedulability tests, to overcome the complexities involved in calculating the response times of multiple instances of CAN messages within the busy period.

2.2. Bit Stuffing in CAN Messages. CAN was designed as a robust and reliable form of communication for short messages. Each data frame carries between 0 and 8 bytes of payload data and has a 15-bit Cyclic Redundancy Check (CRC). The CRC is used by receiving nodes to check for errors in the transmitted message. If a node detects an error in the transmitted message, which may be a bit-stuffing error, a CRC error, a form error in the fixed part of the message or an acknowledgement error, then it transmits an error flag [22]. The error flag consists of 6 bits of the same polarity: "000000" if the node is in the error active state and "111111" if it is error passive. Transmission of an error flag typically causes other nodes to also detect an error, leading to transmission of further error flags.

Figure 1 illustrates CAN error frames, reproduced from [1]. The length of an error frame is between 17 and 31 bits. Hence each message transmission that is signalled as an error

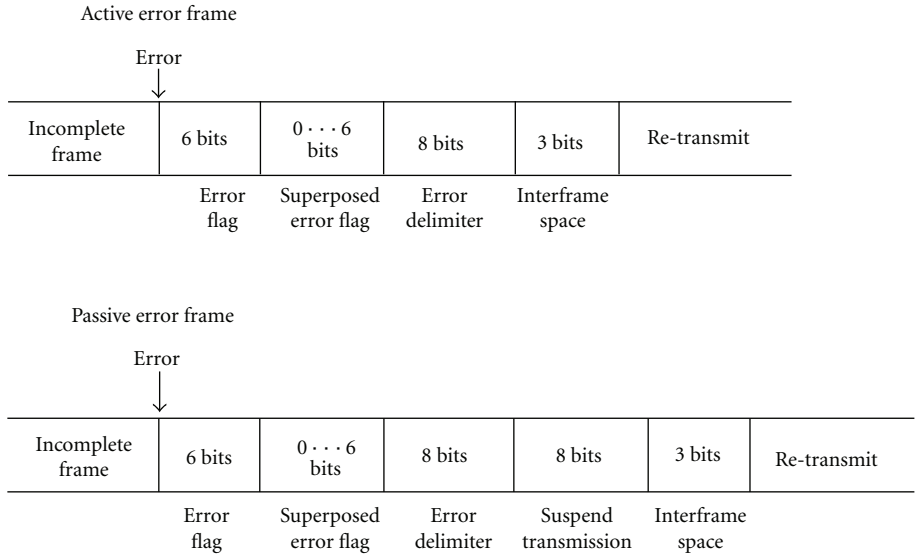

Figure 1: CAN error frames.

can lead to a maximum of 31 additional bits of error recovery overhead plus retransmission of the message itself [22].

One characteristic of Nonreturn-to-Zero code that is adopted in CAN bus is that the signal provides no edges that can be used for resynchronization if transmitting a large number of consecutive bits with the same polarity. Therefore bit stuffing is used to ensure synchronization of all bus nodes. This means that during the transmission of a message, a maximum of five consecutive bits may have the same polarity. The bit-stuffing area in a CAN bus frame includes the SOF, Arbitration field, Control field, Data field, and CRC field. Since bit stuffing is used, six consecutive bits of the same type (111111 or 000000) are considered an error.

As the bit patterns "000000" and "111111" are used to signal errors, it is essential that these bit patterns are avoided in the variable part of a transmitted message (refer to Figure 3). The CAN protocol therefore requires that a bit of the opposite polarity is inserted by the transmitter whenever 5 bits of the same polarity are transmitted. This process referred to as bit stuffing, is reversed by the receiver. The worst-case scenario for bit stuffing is shown in Figure 2 [1]. Note that each stuff bit begins a sequence of 5 bits that is itself subject to bit stuffing.

Stuff bits increase the maximum transmission time of CAN messages. After including stuff bits and the interframe space, the maximum transmission time C_m of a CAN message containing s_m data bytes is given by

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

where g is 34 for standard format (11-bit identifiers) or 54 for extended format (29-bit identifiers), $\lfloor a/b \rfloor$ is notation for the *floor* function, which returns the largest integer less than or equal to a/b, and τ_{bit} is the transmission time for a single bit.

The formula given in (1) simplifies to

$$C_m = (55 + 10s_m)\,\tau_{\text{bit}} \quad (2)$$

for 11-bit identifiers and

$$C_m = (80 + 10s_m)\,\tau_{\text{bit}} \quad (3)$$

for 29-bit identifiers.

2.3. Scheduling Model. The system is assumed to comprise a number of nodes (microprocessors) connected via CAN. Each node is assumed to be capable of ensuring that at any given time when arbitration starts, the highest priority message queued at that node is entered into arbitration [1].

The system is assumed to contain a static set of hard real-time messages each statically assigned to a node on the network. Each message m has a fixed identifier and hence a unique priority. As priority uniquely identifies each message, in the remainder of this paper we will overload m to mean either message m or priority m as appropriate. Each message has a maximum number of data bytes s_m and a maximum transmission time C_m, given by (1).

Each message is assumed to be queued by a software task, process or interrupt handler executing on the host microprocessor. This task is either invoked by, or polls for, the event and takes a bounded amount of time between 0 and J_m to queue the message ready for transmission. J_m is referred to as the *queuing jitter* of the message and is inherited from the overall response time of the task, including any polling delay.

The event that triggers queuing of the message is assumed to occur with a minimum interarrival time of T_m, referred to as the message *period*. This model supports events that occur strictly periodically with a period of T_m, events that occur sporadically with a minimum separation of T_m, and events that occur only once before the system is reset, in which case T_m is infinite.

Each message has a hard deadline D_m, corresponding to the maximum permitted time from occurrence of the initiating event to the end of successful transmission of

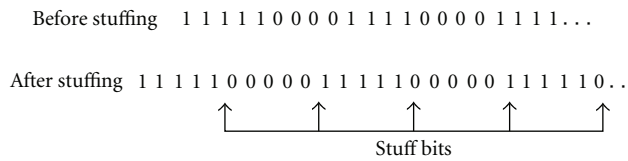

Figure 2: Worst-case bit stuffing.

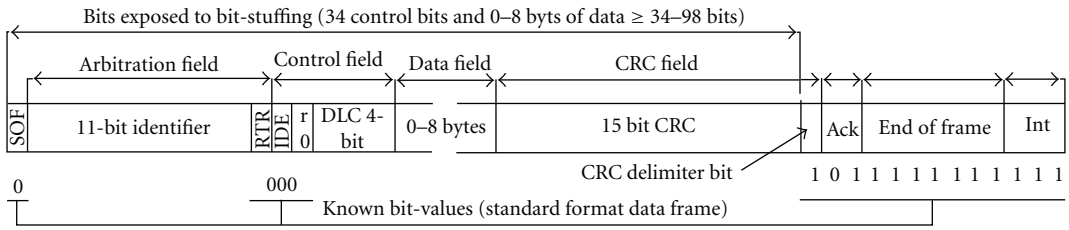

Figure 3: CAN frame layout (standard format data frame).

the message, at which time the message data is assumed to be available on the receiving nodes that require it. Tasks on the receiving nodes may place different timing requirements on the data; however in such cases we assume that D_m is the tightest of such time constraints.

The *worst-case response time* R_m of a message is defined as the longest time from the initiating event occurring to the message being received by the nodes that require it.

A message is said to be *schedulable* if and only if its worst-case response time is less than or equal to its deadline ($R_m \leq D_m$). The system is schedulable if and only if all of the messages in the system are schedulable [1].

2.4. Response Time Analysis. Response time analysis for CAN aims to provide a method of calculating the worst-case response time of each message. These values can then be compared to the message deadlines to determine if the system is schedulable.

For systems complying with the scheduling model given in Section 2.3, the CAN has effectively implemented fixed priority non-pre-emptive scheduling of messages. Following the analysis in [2–5] the worst-case response time of a message can be viewed as being made up of three elements:

(i) the queuing jitter J_m, corresponding to the longest time between the initiating event and the message being queued, ready for transmission on the bus,

(ii) the queuing delay W_m, corresponding to the longest time that the message can remain in the CAN controller slot or device driver queue before commencing successful transmission on the bus,

(iii) the transmission time C_m, corresponding to the longest time that the message can take to be transmitted.

The worst-case response time of message m is given by

$$R_m = J_m + W_m + C_m. \tag{4}$$

The queuing delay comprises *blocking* B_m, due to lower priority messages which may be in the process of being transmitted when message m is queued and *interference* due to higher priority messages which may win arbitration and be transmitted in preference to message m.

The maximum amount of blocking occurs when a lower priority message starts transmission immediately before message m is queued. Message m must wait until the bus is idle before it can be entered into arbitration. The maximum blocking time B_m is given by

$$B_m = \max_{k \in \mathrm{lp}(m)} \left(C_k\right), \tag{5}$$

where $\mathrm{lp}(m)$ is the set of messages with lower priority than m.

The concept of a *busy period*, introduced by Lehoczky [13], is fundamental in analysing worst-case response times. Modifying the definition of a busy period given in [14] to apply to CAN messages, a priority level-m busy period is defined as follows

(i) It starts at some time t^s when a message of priority m or higher is queued ready for transmission, and there are no messages of priority m or higher waiting to be transmitted that were queued strictly before time t^s.

(ii) It is a contiguous interval of time during which any message of priority lower than m is unable to start transmission and win arbitration.

(iii) It ends at the earliest time t^e when the bus becomes idle, ready for the next round of transmission and arbitration, yet there are no messages of priority m or higher waiting to be transmitted that were queued strictly before time t^e.

The key characteristic of a busy period is that all messages of priority m or higher queued strictly before the end of the busy period are transmitted during the busy period. These messages cannot therefore cause any interference on a subsequent instance of message m queued at or after the end of the busy period.

In mathematical terminology, busy periods can be viewed as right half-open intervals: $[t^s, t^e)$ where t^s is the start of

the busy period and t^e the end. Thus the end of one busy period may correspond to the start of another separate busy period. This is in contrast to the simpler definition given in [13], which unifies two adjacent busy periods as we have defined them, and therefore sometimes results in analysis of more message instances than is strictly necessary. For example, in the extreme case of 100% utilisation, the busy period defined in [13] never ends, and an infinite number of message instances would need to be considered.

The worst-case queuing delay for message m occurs for some instances of message m queued within a priority level-m busy period that starts immediately after the longest lower priority message begins transmission. This *maximal* busy period begins with a so-called *critical instant* [1] where message m is queued simultaneously with all higher priority messages, and then each of these messages is subsequently queued again after the shortest possible time intervals. In the remainder of this paper a busy period means this maximum length busy period.

If more than one instance of message m is transmitted during a priority level-m busy period, then it is necessary to determine the response time of each instance in order to find the overall worst-case response time of the message.

In [2–5], Tindell gives the following equation for the worst-case queuing delay:

$$W_m = B_m + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k, \tag{6}$$

where $\mathrm{hp}(m)$ is the set of messages with priorities higher than m and $\lceil a/b \rceil$ is notation for the *ceiling* function which returns the smallest integer greater than or equal to a/b.

Although W_m appears on both sides of (6), as the right hand side is a monotonic nondecreasing function of W_m, the equation may be solved using the following recurrence relation:

$$W_m^n = B_m + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^{n+1} + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k. \tag{7}$$

A suitable starting value is $W_m^0 = B_m$. The relation iterates until either $J_m + W_m^{n+1} + C_m > D_m$, in which case the message is not schedulable or $W_m^{n+1} = W_m^n$, in which case the worst-case response time of the *first instance of the message in the busy period is given by* $J_m + W_m^{n+1} + C_m$.

The flaw in the previous analysis is that, given the constraint $D_m \leq T_m$, it implicitly assumes that if message m is schedulable, then the priority level-m busy period will end at or before T_m. We observe that with fixed priority pre-emptive scheduling this would always be the case, as on completion of transmission of message m, no higher priority message could be awaiting transmission. However, with fixed priority non-pre-emptive scheduling, a higher priority message can be awaiting transmission when message m completes transmission, and thus the busy period can extend beyond T_m [1].

The length t_m of the priority level-m busy period is given by the following recurrence relation, starting with an initial value of $t_m^0 = C_m$ and finishing when $t_m^{n+1} = t_m^n$:

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

where $\mathrm{hp}(m) \cup m$ is the set of messages with priority m or higher. As the right hand side is a monotonic nondecreasing function of t_m, the recurrence relation is guaranteed to converge provided that the bus utilisation U_m, for messages of priority m and higher, is less than 1:

$$U_m = \sum_{\forall k \in \mathrm{hp}(m) \cup m} \frac{C_k}{T_k}. \tag{9}$$

If $t_m \leq T_m - J_m$, then the busy period ends at or before the time at which the second instance of message m is queued. This means that only the first instance of the message is transmitted during the busy period. The existing analysis calculates the worst-case queuing time for this instance via (7) and hence provides the correct worst-case response time in this case.

If $t_m > T_m - J_m$, then the existing analysis may give an optimistic worst-case response time depending upon whether the first or some subsequent instance of message m in the busy period has the longest response time.

The analysis presented in Appendix A.2 of [15] suggests that t_m is the smallest value that is a solution to (8); however this is not strictly correct [1]. For the lowest priority message, $B_m = 0$ and so $t_m = 0$ is trivially the smallest solution. This problem can be avoided by using an initial value of $t_{\mathrm{m}}^0 = C_m$ [1].

The number of instances Q_m of message m that become ready for transmission before the end of the busy period is given by

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil. \tag{10}$$

To determine the worst-case response time of message m, it is necessary to calculate the response time of each of the Q_m instances and then take the maximum of these values.

In the following analysis, the index variable q is used to represent an instance of message m. The first instance in the busy period corresponds to $q = 0$ and the final instance to $q = Q_m - 1$. The longest time from the start of the busy period to the instance at q beginning successful transmission is given by

$$W_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k. \tag{11}$$

The recurrence relation starts with a value of $W_m^0(q) = B_m + qC_m$ and ends when $W_m^{n+1}(q) = W_m^n(q)$ or when $J_m + W_m^{n+1}(q) - q\mathrm{T}_m + C_m > D_m$ in which case the message is unschedulable. For values of $q > 0$ an efficient starting value

is given by $W_m^0(q) = \mathrm{W}_m(q-1) + C_m$. The event of initiating instance q of the message occurs at time $qT_m - J_m$ relative to the start of the busy period, so the response time of instance q is given by

$$R_m(q) = J_m + W_m(q) - qT_m + C_m. \tag{12}$$

The worst-case response time of message m is therefore

$$R_m = \max_{q=0\cdots Q_m-1} \left(R_m(q)\right). \tag{13}$$

The analysis presented previously is also applicable when messages have deadlines that are greater than their periods, so-called arbitrary deadlines [1]. However, if such timing characteristics are specified, then the software device drivers or CAN controller hardware may need to be capable of buffering more than one instance of a message. The number of instances of each message that need to be buffered is bounded by

$$N_m = \left\lceil \frac{R_m}{T_m} \right\rceil. \tag{14}$$

The analysis presented in [15] effectively uses $Q_m = \lfloor t_m/T_m \rfloor + 1$ rather than $Q_m = \lceil t_m/T_m \rceil$. This yields a value which is one too large when the length of the busy period plus jitter is an integer multiple of the message period. Although this does not give rise to problems, the more efficient formulation given by (10) is preferred [1].

The analysis given in this section as per Davis et al. [1] corrects a significant flaw in the previous schedulability analysis for CAN, given by Tindell et al. [2–5]. However, this schedulability test presented is more complex, potentially requiring the computation of multiple response times.

An upper bound on the queuing delay of the second and subsequent instances of message m within the busy period is therefore given by

$$W_m^{n+1} = C_m + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^n + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k. \tag{15}$$

This result suggests a simple but pessimistic schedulability test. An instance of message m can either be subject to blocking due to lower priority messages or to push through interference of at most C_m due to the previous instance of the same message, but not both. Hence we can modify (7) to provide a correct sufficient but not necessary schedulability test:

$$W_m^{n+1} = \max(B_m, C_m) + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^n + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k. \tag{16}$$

A further simplification is to assume that the blocking factor always takes its maximum possible value:

$$W_m^{n+1} = B^{\max} + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^n + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k, \tag{17}$$

where $B^{\max}$ corresponds to the transmission time of the longest possible CAN message (8 data bytes) irrespective of the characteristics and priorities of the messages in the system. So far we have assumed that no errors occur on the CAN bus. However as originally shown in [2–5], schedulability analysis of CAN may be extended to include an appropriate error model.

In [1] it is assumed that the maximum number of errors present on the bus in some time interval $[0, t]$ is given by the function $F(t)$. No specific detail about this function is assumed, save that it is a monotonic non-decreasing function of t. The schedulability equations are modified to account for the error recovery overhead. The worst-case impact of a single bit error is to cause transmission of an additional 31 bits of error recovery overhead plus retransmission of the affected message. Only errors affecting message m or higher priority messages can delay message m from being successfully transmitted. The maximum additional delay caused by the error recovery mechanism is therefore given by

$$E_m(t) = \left(31\tau_{\mathrm{bit}} + \max_{k \in \mathrm{hp}(m)\mathrm{u}_m} \left(C_k\right)\right) F(t). \tag{18}$$

Revising (8) to compute the length of the busy period we have

$$t_m^{n+1} = E_m\left(t_m^n\right) + B_m + \sum_{\forall k \in \mathrm{hp}(m)\mathrm{u}_m} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k. \tag{19}$$

Again an appropriate initial value is $t_m^0 = C_m$. Equation (19) is guaranteed to converge, provided that the utilisation U_m including error recovery overhead is less than 1.

As before, (10) can be used to compute the number of message instances that need to be examined to find the worst-case response time:

$$W_m^{n+1}(q) = E_m\left(W_m^n + C_m\right) + B_m + qC_m + \sum_{\forall k \in \mathrm{hp}(m)} \left\lceil \frac{W_m^n + J_k + \tau_{\mathrm{bit}}}{T_k} \right\rceil C_k. \tag{20}$$

Equation (20) extends (11) to account for the error recovery overhead. Note that as errors can impact the transmission of message m itself, the time interval considered in calculating the error recovery overhead includes the transmission time of message m as well as the queuing delay. Equations (20), (12), and (13) can be used together to compute the response time of each message instance q and hence find the worst-case response time of each message in the presence of errors at the maximum rate specified by the error model.

The sufficient schedulability tests given earlier in this section can be similarly modified via the addition of the term $E_m(W_m^n + C_m)$ to account for the error recovery overhead [1].

3. Probabilistic Response Time Analysis of CAN Messages

3.1. Probabilistic Bit-Stuffing Distributions. When performing worst-case response-time analysis, the worst-case number of stuff bits is traditionally used. In [7], Nolte et al. introduce a worst-case response time analysis method which uses distributions of stuff bits instead of the worst-case values. This makes the analysis less pessimistic, in the sense that we obtain a distribution of worst-case response times corresponding to all possible combinations of stuff bits of all message frames involved in the response time analysis. Using a distribution rather than a fixed value makes it possible to select a worst-case response time based on a desired probability p of violation; that is, the selected worst-case response time is such that the probability of a response-time exceeding it is $\leq p$. The main motivation for calculating such probabilistic response-times is that they allow us to reason about tradeoffs between reliability and timeliness.

The number of bits, apart from the data part in the frame, which are exposed to the bit-stuffing mechanism, is defined as g which is in the range $\{34, 54\}$. This is because we have either 34 (CAN standard format) or 54 (CAN extended format) bits which are exposed to the bit-stuffing mechanism. 10 bits in the CAN frame are not exposed to the bit-stuffing mechanism (refer to Figure 3). The number of bytes of data in CAN message frame i is defined as L_i which is in the range $[0, 8]$.

Recall that a CAN message frame can contain 0 to 8 bytes of data. According to the CAN standard [22], the total number of bits in a CAN frame before bit stuffing is therefore

$$8L_i + g + 10, \tag{21}$$

where 10 is the number of bits in the CAN frame not exposed to the bit-stuffing mechanism. Since only $g + 8L_i$ bits in the CAN frame are subject to bit stuffing, the total number of bits after bit stuffing can be no more than

$$8L_i + g + 10 + \left\lfloor \frac{g + 8L_i - 1}{4} \right\rfloor. \tag{22}$$

Intuitively the above formula captures the number of stuffed bits in the worst case scenario, shown in Figure 2.

The expression (22) describes the length of a CAN frame in the worst case. In [6], the number of stuff bits is represented as a distribution. By using a distribution of stuff bits instead of the worst-case number of stuff bits, it is possible to obtain a distribution of response times that allow to calculate less pessimistic (compared to traditional worst-case) response times based on probability.

Firstly, let us define γ as the distribution of stuff bits in a CAN message frame. We express γ as a set of pairs containing the number of stuff bits with corresponding probability of occurrence. Each pair is defined as $(x, P(x))\varepsilon\gamma$, where $P(x)$ is the probability of exactly x stuff bits in the CAN frame. Note that $\sum_{x=0}^{\infty} P(x) = 1$.

As shown in [6], we can extract 9 different distributions of stuff bits depending on the number of bytes of data in the CAN message frame. We define γ_{L_i} as the distribution representing a CAN frame containing L_i bytes of data. Recall that L_i is the number of bytes of data (0 to 8) in a message frame i.

We define $n = \gamma(p)$ as the worst-case number of stuff bits, n, to expect with a probability P based on the stuff-bit distribution γ, that is, $\sum_{x=n+1}^{\infty} P(x) \leq p$, or to express it in another way, the probability of finding more than n stuff bits, based on the stuff-bit distribution γ, is $\leq p$.

Note that the selection of a probability P should be done based on the requirements of the application. With a proper value for p, the worst case mean time to failure should sufficiently exceed what is required. Finally, by assuming (as in [6]) that CAN message frames are independent in the sense of number of stuff bits, we can define $\prod_n \gamma$ as the joint distribution corresponding to the combination of n distributions of stuff bits; that is, the number of stuff bits caused by a sequence of n messages sent on the bus is described by $\prod_n \gamma = \gamma x \gamma x \gamma \cdots x \gamma$,

where x denotes multiplicative combination of discrete distributions. If the distributions happen to be equal, $\prod_n^{-} \gamma$ is defined as the joint distribution of n *equal* distributions of stuff bits; that is, the number of data bytes is the same for all messages considered by the expression.

In order to include the bit-stuffing distributions in (12), we need to redefine C_i and B_i as $C_i(p)$ and $B_i(p)$, where

$$C_i(p) = c_i + \gamma_{L_i}(p)\,\tau_{\text{bit}}, \tag{23}$$

where γ_{L_i} is the distribution of stuff-bits in the message and C_i is the transmission time of message i excluding stuff-bits:

$$\begin{aligned} C_i &= \left(8L_i + g + 10\right)\tau_{\text{bit}}, \\ B_i(p) &= b_i + \gamma_{\max_{k\in \text{lp}(i)}(L_k)}(p)\,\tau_{\text{bit}}, \\ b_i &= \max_{k\in \text{lp}(i)}\left(C_k\right) + 3\tau_{\text{bit}}, \\ R_i^n(p) &= J_i + b_i + C_i \\ &\quad + \sum_{j\in \text{hp}(i)} I_j\left(R_i^{n-1}(p) - J_i - C_i\right)\left(C_j + 3\tau_{\text{bit}}\right) \\ &\quad + \psi_i(p)\,\tau_{\text{bit}}, \end{aligned} \tag{24}$$

where ψ_i is defined as the distribution of the total number of stuff-bits of all messages involved in the response time analysis for message i.

This approach obtains the maximum stuffed bits under a given probability P, to reduce pessimism of the worst-case response time and busload value.

Anyu Cheng et al. in [23] extend this work in [7] and gives the probability distribution curves of stuffed bits in message's different lengths by introducing the probability model of stuffed bits. They design and develop scheduling analysis software on fixed priority message scheduling. Then they use the software to analyses the schedulability for the messages in a hybrid electric vehicle. Furthermore, a simulation experiment based on CANoe was made to test the design. By comparing the results, it shows that algorithm based on the probability model of stuffed bits is right, and the designed software is accurate and reliable.

3.2. Probabilistic Error Model. The analysis as presented does not cover the effect of transmission errors. Obviously, detected errors trigger the transmission of an error frame as well as a retransmission which increases the busy window and therefore the response time. On the other hand a longer busy window might increase the probability that successive errors might affect the busy window [24]. In order to include effects of errors (e.g., retransmission overhead) different approaches were introduced.

3.2.1. Related Work. A method to analyse worst-case real-time behaviour of a CAN bus was developed by Tindell et al. [5]. By applying processor scheduling analysis to the CAN bus, they showed that in the absence of faults the worst-case response time of any message is bounded and can be accurately predicted. Moreover, the analysis can be extended in order to handle the effect of errors in the channel.

The error recovery mechanism of CAN involves the retransmission at any corrupted messages. An additional term can be introduced into their analysis, called the error recovery overhead function, which is the upper bound of the overhead caused by such retransmissions in a time interval. A very simple fault model is used [5], to show how the schedulability analysis is performed in the presence of errors in the channel. The model is based on a minimum interarrival time between faults. The authors note that the error recovery function can be more accurately determined either from observation of the behaviour of CAN under high noise conditions or by building a statistical model.

Punnekkat et al. [8] extend the work of Tindell et al. by providing a more general fault model which can deal with interference caused by several sources. Punnekkat's model assumes that every source of interference has a specific pattern, consisting of an initial burst of errors and then a distribution of faults with a known minimum interarrival time. Except for the more general fault model, the rest of the schedulability analysis is performed like [5].

Both Tindell and Punnekkat use models based on a minimum interarrival time between faults and therefore assume that the number of faults that can occur in an interval is bounded. In the environment where CAN is used, faults are caused mainly by Electromagnetic Interference (EMI) which is often observed as a random pulse train with a Poisson distribution [24]. Therefore the assumption made by the bounded model may not be appropriate for many systems because there is a realistic probability of faults occurring closer than the minimum interarrival time.

Unlike Tindell and Punnekkat, Navet et al. [25] propose a probabilistic fault model, which incorporates the uncertainty of faults caused by EMI. The fault model suggested by Navet uses a stochastic process which considers both the frequency of the faults and their gravity. In that model, faults in the channel occur according to a Poisson law and can be either single-bit faults or burst errors (which have a duration of more than one bit) according to a random distribution. This allows the interference caused by faults in the channel to be modeled as a generalised Poisson process. Note that if the occurrence of faults in the channel follows a Poisson law, the maximum number of transmission errors suffered by the system in a given interval is not bounded, so the probability of having sufficient interference to prevent a message from meeting its deadline is always nonzero; therefore every system is inherently unschedulable.

Hence Navet's analysis does not try to determine whether a system is schedulable (as [5, 8]), but it calculates the probability that a message does not meet its deadline. Obtaining such a probability, named Worst Case Deadline Failure Probability (WCDFP), gives a measure of the system reliability, because a lower value of the WCDFP implies a high resilience to interference.

Navet's analysis uses the scheduling analysis of Tindell to calculate the maximum number of faults that can be tolerated for each message before the deadline is reached. This number is called K_m and only depends on the characteristics (length, priority, period, etc.) of the message set. The worst-case response time that K_m faults would generate is called $R_{m\,\max}$. Once K_m and $R_{m\,\max}$ are obtained, they are used with the fault model to find the probability that a message may miss its deadline. Navet defines the WCDFP of a message m as the probability that more than K_{rn} errors occur during $R_{m\,\max}$. This probability can be analytically calculated as the fault model assumed by Navet is a generalized Poisson process.

The main drawback of the analysis is that it includes two inaccuracies which increase the pessimism in the estimation of the WCDFP. The first source of pessimism is implicit in the definition of WCDFP. The definition of WCDFP does not properly reflect the conditions in which a message can miss its deadline. In order for a message to miss a deadline, faults in the channel is required to occur while the message is queued or in transmission; a fault occurring after the message has been received cannot delay the message. This condition is more restrictive than the condition used in [25], which is that K_m errors occur at any time during the maximum response time of the message, independently of whether the message has already been received.

The second source of pessimism is an overly pessimistic assumption about the nature of burst errors where a fault causes a sequence of bits to be corrupted. In Navel's analysis, a burst error of duration "u" bits is treated as a sequence of u single bit faults [25, Equation (7)], each causing a maximal error overhead (an error frame and the retransmission of a frame of higher or equal priority). This assumption is inconsistent with the CAN protocol specification [22] since in reality a burst error can cause retransmission of only one frame, because no message is sent again until the effect of the burst is finished. This causes pessimism of several orders of magnitude.

A different method to calculate probability of deadline failure in CAN under fault conditions is proposed in [9]. This work points out that errors happening during bus idle do not cause any message retransmission, and therefore those errors cause interference lower than the interference typically considered in scheduling analysis. To avoid this source of pessimism when performing scheduling analysis, the effect of errors is modelled with a fixed pattern of interference; this is a simplification of the fault model presented in [8]. Due to this determinism, interactions between messages and errors can be analysed through simulation, and then the probability of

having a message that misses its deadline can be determined. Nevertheless, this method has important drawbacks. First, an interference pattern for every possible error source is hard to be determined. And second, combination of several error sources increases the complexity of the analysis to such an extent that it becomes infeasible, so random sampling is used.

Modelling arrivals of errors with a random distribution, as done in [10], allow a more generalized solution. Broster et al. [26] propose an analysis that provides an accurate probability of deadline failure without excess pessimism, based on the assumption that faults are randomly distributed.

In [27], an approach is presented to tightly bound the reliability for periodic, synchronized messages. Therefore, a reliability metric $R(t)$ is defined which denotes the probability that CAN communication survives time t without a deadline miss. The reliability is calculated based on the hyperperiod, which is the time when the activation pattern of a periodic message set repeats itself. It is defined by the least common multiple over all periods. Hence, the complexity of the algorithm depends on the amount of activations in the hyperperiod. This algorithm is suitable for automotive message sets in which periods are typically multiples of 10 ms. However, if messages are not synchronized, or the relative phasing is unknown, the approach is not applicable. In [26], the busy-window approach is used, and a tree-based approach is presented, where different error scenarios are evaluated iteratively. In a second step, these scenarios are translated to probabilities and a worst-case deadline failure probability is calculated. The approach was extended in [28], and the tree-based was superseded by a simpler, more accurate approach. However, both methods [26, 28] allow only deadlines smaller than the periods, which is a limit for practical use since bursty CAN traffic is not supported. In [24], existing methods are generalized to support arbitrary deadlines and derive a probabilistic response time bound.

3.2.2. Error Model. In [24, 26] the occurrence of errors is modeled by using a Poisson model. Practically, a Poisson process models independent single bit errors (without bursts), where λ specifies the bit error rate. The probability for the occurrence of m error-events in the time window λt is

$$p\left(m, \Delta t\right) n = \frac{e^{-\lambda \Delta t} (\lambda \Delta t)^m}{m!}. \tag{25}$$

It is possible that a message of length C is hit by multiple error events and only one retransmission occurs (e.g., after reception when the CRC is checked), but it is assumed that in the worst-case condition, each error event will lead to exactly one retransmission. Thus, we can directly use (25) to obtain the probability that K error events occur during a given time window, and the probability for the error-free case is

$$P\left(w_{i|0}\right) = p\left(0, w_{i|0}\right) = \mathrm{e}^{-\lambda_{w_{i|0}}}. \tag{26}$$

For $K > 0$, it is not enough to just calculate $p(K, w_{i|K})$, because error events have to occur in certain segments of the busy window, and more efficient technique was used in [27], which can be applied for the general case in which a busy-window includes multiple queued activations which can be affected by errors. The approach works as follows: one error-event in the entire busy window $w_{i|1}$ can happen in two ways. The error may actually lead to an $w_{i|1}$ busy window with the probability $P(w_{i|1})$. Or, we face a busy window of length $w_{i|0}$ and the error event occurs in the interval $(w_{i|0}, w_{i|1})$:

$$p\left(1, w_{i|1}\right) = P\left(w_{i|1}\right) + P\left(w_{i|0}\right) p\left(1, w_{i|1} - w_{i|0}\right). \tag{27}$$

The value of $P(w_{i|1})$ can then be obtained by rearranging the equation. Similarly we can apply this idea to $K = 2$. Two errors in the time window $w_{i|2}$ may occur in the following mutually exclusive ways. (i) A busy window of length $w_{i|2}$ actually occurs assuming two error events with the probability $P(w_{i|2})$. (ii) $w_{i|1}$, occurred which implies exactly one error in $w_{i|1}$ and the second error must then happen in the interval $(w_{i|1}; w_{i|2})$. (iii) $w_{i|0}$ occurred which implies no error in $w_{i|0}$. And exactly two errors must be in the interval $(w_{i|0}, w_{i|2})$:

$$\begin{aligned} p\left(2, w_{i|2}\right) = {} & P\left(w_{i|2}\right) + P\left(w_{i|1}\right) p\left(1, w_{i|2} - w_{i|1}\right) \\ & + P\left(w_{i|0}\right) p\left(2, w_{i|2} - w_{i|0}\right). \end{aligned} \tag{28}$$

By rearranging the equation for $P(w_{i|2})$, we get the probability for a $K = 2$ busy window. The same argument is valid for the following K-error busy windows, and (28) is generalized into the following form:

$$\begin{aligned} P\left(w_{i|K}\right) = {} & p\left(K, w_{i|K}\right) \\ & - \sum_{j=0}^{K-1} P\left(w_{i|j}\right) p\left(K - j, w_{i|K} - w_{i|j}\right). \end{aligned} \tag{29}$$

The worst-case response time exceedance function can be calculated as

$$P^{+}\left[R_i > r\right] = 1 - \sum_{\forall K | R_{i|K} < r} P\left(w_{i|K}\right). \tag{30}$$

Practically, this function denotes a bound for the probability that a response time exceeds a certain threshold, and the probability that a deadline is exceeded can be bounded to $P^{+}[R_i > D_i]$.

4. Conclusion

In this review paper, the worst case response time analysis of messages in controller area network and the probabilistic response time analysis of CAN messages are reviewed. The worst-case response time analysis includes the worst-case response time analysis presented in early 1990s by Tindell et al. [2–5] and the worst case response time analysis by Davis et al. [1] in 2007. Davis et al. in [1] have pointed out the flaw in the earlier analysis by Tindel et al. and showed that multiple instances of the CAN messages should be analysed to determine the response time and hence the schedulability of the CAN messages. The worst-case response time analysis leads to excessive level of pessimism; we may choose a pessimistic approach but with as little pessimism as possible,

since worst case does not always occur. The probabilistic response time analysis of CAN messages is recommended; here two approaches are considered [6, 7], namely, instead of using the worst-case bit-stuffing pattern, we can consider a distribution of possible bit stuffing according to the application and select one most probable bit-stuffing pattern, thereby we are less pessimistic; another probabilistic approach is considering the probability of occurrence of errors [24–26]. In worst-case analysis, it is assumed that every error flag transmitted has a retransmission associated, whereas this is not true, since the same error can cause many error flags and only one retransmission. This assumption causes some level of pessimism. There are different methods presented in [6] whereby we can reduce the number of stuff bits, either by using XOR operation on the messages before transmission (encoding) and redoing the XOR after reception (decoding), thus avoiding having continuous bits of zeros or ones, thereby avoiding bit stuffing. The other method presented in [6] is to choose the priorities such that the identifier bits do not have continuous ones or zeros, thereby avoiding bit stuffing. Of course in this method the number of priorities that can be used is reduced.

Another approach in making the best usage of the bandwidth is to schedule the messages with offsets, which leads to a desynchronization of the message streams. This "traffic shaping" strategy is very beneficial in terms of worst-case response times [29, 30]. The Worst-Case Response Time (WCRT) for a frame corresponds to the scenario where all higher priority CAN messages are released synchronously. Avoiding this situation and thus reducing WCRT can be achieved by scheduling stream of messages with offsets. Precisely, the first instance of a stream of periodic frames is released with a delay, called the offset, in regard to a reference point which is the first time at which the station is ready to transmit. Subsequent frames of the streams are then sent periodically, with the first transmission as time origin. The choice made for the offset values has an influence on the WCRT, and the challenge is to set the offsets in such a way so as to minimize the WCRT, which involves spreading the workload over time as much as possible. The future work is to present the review of statistical approach to response time analysis. It is proposed that a fusion of methods may be adopted to cater to the requirement of the application; for safety critical application like automotive and industrial application, the worst-case response time analysis is recommended, and for noncritical applications where we can introduce some tolerance we may apply the probabilistic response time analysis.

References

[1] 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, no. 3, pp. 239–272, 2007.

[2] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: controller area network (CAN)," in *Proceedings of the 15th IEEE Real-Time Systems Symposium (RTSS '94)*, pp. 259–263, IEEE Computer Society Press, 1994.

[3] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

[4] K. Tindell and A. Burns, "Guaranteeing message latencies on control area network (can)," in *Proceedings of the 1st International CAN Conference*, Citeseer, 1994.

[5] K. 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] T. Nolte, H. Hansson, and C. Norstrom, "Minimizing CAN response-time analysis jitter by message manipulation," in *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '02)*, pp. 197–206, September 2002.

[7] T. Nolte, H. Hansson, and C. Norstrom, "Probabilistic worst-case response-time analysis for the controller area network," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '03)*, pp. 200–207, May 2003.

[8] S. Punnekkat, H. Hansson, and C. Norstrom, "Response time analysis under errors for CAN," in *Proceedings of the 6th IEEE Real-Time Technology and Applications Symposium (RTAS '00)*, pp. 258–265, June 2000.

[9] H. Hansson, C. Norström, and S. Punnekkat, "Integrating reliability and timing analysis of CAN-based systems," in *Proceedings of the IEEE International Workshop on Factory Communication Systems (WFCS '00)*, IEEE Industrial Electronics Society, Porto, Portugal, September 2000.

[10] H. Hansson, C. Norström, and S. Punnekkat, "Reliability modelling of time-critical distributed systems," in *Proceedings of the 6th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT '00)*, M. Joseph, Ed., vol. 1926 of *Lecture Notes in Computer Science*, Springer, Pune, India, September 2000.

[11] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat, "Using bit-stuffing distributions in CAN analysis," in *Proceedings of the IEEE/IEE Real-Time Embedded Systems Workshop (RTES '01)*, December 2001.

[12] K. W. Tindell, A. Burns, and A. J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, vol. 6, no. 2, pp. 133–151, 1994.

[13] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proceedings of the 11th IEEE Real-Time Systems Symposium (RTSS '90)*, pp. 201–209, IEEE Computer Society Press, December 1990.

[14] M. G. Harbour, M. H. Klein, and J. P. Lehoczky, "Fixed priority scheduling periodic tasks with varying execution priority," in *Proceedings of the 12th IEEE Real-Time Systems Symposium (RTSS '91)*, pp. 116–128, IEEE Computer Society Press, December 1991.

[15] L. George, N. Rivierre, and M. Spuri, "Pre-emptive and non-pre-emptive real-time uni-processor scheduling," Tech. Rep. 2966, Institut National de Recherche et Informatique et en Automatique, Versailles, France, 1996.

[16] R. J. Bril, "Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred pre-emption is too optimistic," CS-Report 06-05, Technische Universiteit, Eindhoven, The Netherlands, 2006.

[17] A. Burns, "Pre-emptive priority based scheduling: an appropriate engineering approach," in *Advances in Real-Time Systems*, S. Son, Ed., pp. 225–248, Prentice-Hall, 1994.

[18] Y. Wang and M. Saksena, "Scheduling fixed priority tasks with pre-emption threshold," in *Proceedings of the 6th International Workshop on Real-Time Computing Systems and Applications (RTCSA '99)*, pp. 328–335, December 1999.

[19] J. Regehr, "Scheduling tasks with mixed pre-emption relations for robustness to timing faults," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS '02)*, pp. 315–326, IEEE Computer Society Press, December 2002.

[20] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (CAN) refuted," CS-Report 06-19, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2006.

[21] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (CAN) refuted," in *Proceedings of the 5th International Workshop on Real-Time Networks (RTN '06)*, 2006.

[22] International Standards Organisation, *ISO 11898. Road Vehicles—Interchange of Digital Information—Controller Area Network (CAN) for High-Speed Communication*, 1993.

[23] A. Cheng, L. Zhang, and T. Zheng, "The schedulability analysis and software design for networked control systems of vehicle based on CAN," in *Proceedings of the IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE '11)*, vol. 2, pp. 274–278, Wuhan, China, August 2011.

[24] P. Axer, M. Sebastian, and R. Ernst, "Probabilistic response time bound for CAN messages with arbitrary deadlines," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE '12)*, pp. 1114–1117, Dresden, Germany, March 2012.

[25] N. Navet, Y. Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *Journal of Systems Architecture*, vol. 46, no. 7, pp. 607–617, 2000.

[26] I. Broster, A. Burns, and G. Rodriguez-Navas, "Probabilistic analysis of can with faults," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS '02)*, pp. 269–278, 2002.

[27] M. Sebastian and R. Ernst, "Reliability analysis of single bus communication with real-time requirements," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC '09)*, pp. 3–10, November 2009.

[28] I. Broster and A. Burns, "Comparing real-time communication under electromagnetic interference," in *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS '04)*, pp. 45–52, July 2004.

[29] M. Grenier, L. Havet, and N. Navet, "Pushing the limits of CAN—scheduling frames with offsets provides a major performance boost," in *Proceedings of the 4th European Congress on Embedded Real Time Software*, Toulouse, France, 2008.

[30] L. Du and G. Xu, "Worst case response time analysis for CAN messages with offsets," in *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES '09)*, pp. 41–45, Pune, India, November 2009.