Probabilistic Worst-Case Response-Time Analysis for the Controller Area Network

Thomas Nolte, Hans Hansson, and Christer Norström
Mälardalen Real-Time Research Centre
Department of Computer Science and Engineering
Mälardalen University, Västerås, SWEDEN
http://www.mrtc.mdh.se

Abstract

This paper presents a novel approach for calculating a probabilistic worst-case response-time for messages in the Controller Area Network (CAN). CAN uses a bit-stuffing mechanism to exclude forbidden bit-patterns within a message frame. The added bits eliminate the forbidden patterns but cause an increase in frame length. How much the length is increased depends on the bit-pattern of the original message frame.

Traditional response-time analysis methods assume that all frames have a worst-case number of stuff-bits. This introduces pessimism in the analysis.

In this paper we introduce an analysis approach based on using probability distributions to model the number of stuff-bits. The new analysis additionally opens up for making trade-offs between reliability and timeliness, in the sense that the analysis will provide a certain probability for missing deadlines, which in the reliability analysis can be treated as a probability of failure. We evaluate the performance of our method using a subset of the SAE[1] *benchmark.*

1. Introduction

During the last decade 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 [2, 4, 12, 16]. 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.

[1]See [11] for details.

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

This separation of deterministic (0/1) schedulability analysis and stochastic reliability analysis is a natural simplification of the total analysis. However the deterministic schedulability analysis is unfortunately quite pessimistic, since it assumes that a missed deadline in the worst case is equivalent to always missing the deadline. There are also other sources of pessimism in the analysis, including considering worst-case execution times and the usage of pessimistic fault models.

Reliability is defined as the probability that a system can perform its intended function, under given conditions, for a given time interval. A major issue is how to compose hardware reliability, software reliability, environmental model, and timely correctness to arrive at reasonable estimates of overall system reliability, as depicted in Figure 1.

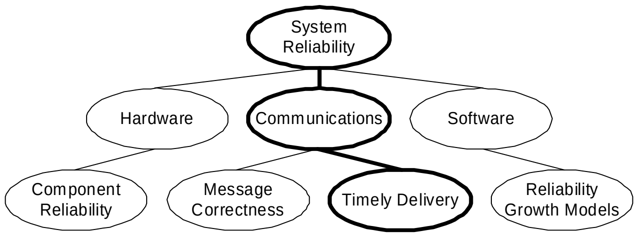

Figure 1. System reliability: a top-down view.

The Controller Area Network is extensively used in small scale distributed systems, such as automotive, medical and industrial applications. In this paper we provide a probabilistic response-time analysis method for messages in

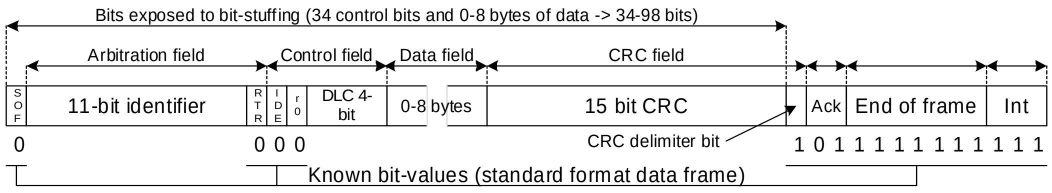

Figure 2. CAN frame layout (standard format data frame).

the Controller Area Network (CAN). Several probabilistic approaches for schedulability analysis of real-time systems have been presented, e.g., [1, 7]. However, none of these specifically address CAN.

We have in our previous work presented a method to model the number of *stuff-bits* in a CAN message frame [8, 9]. Stuff-bits are extra bits added by the CAN protocol. There is a built in mechanism in the CAN protocol which removes forbidden bit-patterns (e.g., patterns used for error signalling and the communication protocol) within the message frame by “inserting” stuff-bits at specific positions. This mechanism causes a variation in the CAN message frame length.

When performing worst-case response-time analysis, the worst case number of stuff-bits is traditionally used. In this paper we will 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, i.e., the selected worst-case response time is such that the probability of a response-time exceeding it is $\leq p$. Our main motivation for calculating such probabilistic response-times is that they allow us to reason about trade-offs between reliability and timeliness.

However, it should be noted that this paper focuses on a single aspect, namely a probabilistic *worst-case response time*, based on using bit-stuffing distributions. There are other parameters, including execution times and phasings of message queuings, that have similar variations and effects on the response-time analysis. However, our calculations are based on the “critical instant” worst-case scenario.

Outline: Section 2 presents the traditional schedulability analysis for CAN. In Section 3 we present the new probabilistic response-time analysis, and in Section 4 the analysis is evaluated using the SAE [11] benchmark. Finally Section 5 concludes the paper and presents some future work.

2. Traditional Schedulability Analysis of CAN Frames

The Controller Area Network (CAN) [10] is a broadcast bus designed to operate at speeds of up to 1 Mbps. Data is transmitted in frames containing between 0 and 8 bytes of data and a number of control bits. Depending on the CAN format (standard or extended) the number of control bits are either 44 or 64. Between CAN frames sent on the bus, there is also a 3 bit inter-frame space. The standard format CAN frame (and the inter-frame space) is shown in Figure 2.

The difference between the standard and the extended format is that the extended format has 29 identifier bits instead of the 11 bits used in the standard format. The identifier is required to be unique, in the sense that two simultaneously active frames originating from different sources (i.e., nodes or CAN-controllers) must have distinct identifiers. The identifier serves two purposes: (1) assigning a priority to the frame, and (2) enabling receivers to filter frames. For a more detailed explanation of the different fields in the CAN frame, please consult [10, 5].

CAN is a collision-avoidance broadcast bus, which uses deterministic collision resolution to control access to the bus (so called CSMA/CA). The basis for the access mechanism is the electrical characteristics of a CAN bus: if multiple stations are transmitting concurrently and one station transmits a ‘0’ then all stations monitoring the bus will see a ‘0’. Conversely, only if all stations transmit a ‘1’ will all processors monitoring the bus see a ‘1’. During arbitration, competing stations are simultaneously putting their identifiers, one bit at the time, on the bus. By monitoring the resulting bus value, a station detects if there is a competing higher priority frame and stops transmission if this is the case. Because identifiers are unique within the system, a station transmitting the last bit of the identifier without detecting a higher priority frame must be transmitting the highest priority queued frame, and hence can start transmitting the body of the frame.

2.1. Classical CAN Bus Analysis

Tindell et al. [13, 14, 15] present analysis to calculate the worst-case latencies of CAN frames. This analysis is based on the standard fixed priority response time analysis for CPU scheduling [2].

Calculating the response times requires a bounded worst case queuing pattern of frames. The standard way of expressing this is to assume a set of traffic streams, each generating frames with a fixed priority. The worst-case behaviour of each stream, in terms of network load, is to send as many frames as they are allowed, i.e., to periodically queue frames. In analogue with CPU scheduling, we obtain a model with a set $\mathcal{S}$ of streams (corresponding to CPU tasks). Each $S_i \in \mathcal{S}$ is a triple $< P_i, T_i, C_i >$, where P_i is the priority (defined by the message frame identifier), T_i is the period and C_i the worst-case transmission time of frames sent on stream S_i. The worst-case latency R_i of a CAN frame sent on stream S_i is, if we assume the minimum variation in queuing time relative to T_i to be 0, defined by

$$R_i = J_i + q_i + C_i \tag{1}$$

where J_i is the queuing jitter of the frame, i.e., the maximum variation in queuing time relative start of T_i, inherited from the sender task which queues the frame, and q_i represents the effective queuing time, given by

$$q_i^n = B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil (C_j + 3\tau_{bit}) \tag{2}$$

where

- $B_i = \max_{k \in lp(i)} (C_k) + 3\tau_{bit}$ is the worst-case blocking time of frames sent on S_i, where $lp(i)$ is the set of streams with priority lower than S_i. The reason for the blocking factor is that transmissions are non preemptive, i.e., after bus arbitration has started the frame with the highest priority among competing frames will be transmitted until completion, even if a frame with higher priority gets queued before the transmission is completed.

- $hp(i)$ is the set of streams with priority higher than S_i.

- τ_{bit} (the bit-time) caters for the difference in arbitration start times at the different nodes due to propagation delays and protocol tolerances.

- C_j is the transmission time of message j. How to calculate C_j is presented in the next section.

- $3\tau_{bit}$ represents the inter-frame space (traditionally [13, 14, 15], the inter-frame space was considered a part of the data frame, but separating it [3] removes a small source of pessimism in the equations).

Note that Equation 2 is a recurrence relation, where the approximation to the $(n+1)$th value is found in terms of the nth approximation, with the first approximation set to $q_i^0 = 0$. A solution is reached either when the $(n+1)$th value is equal to the nth, or when R_i exceeds its message deadline or period. The recurrence relation will terminate given that the total bus utilization is ≤ 1, i.e., $\sum_{S_i \in \mathcal{S}} \left(\frac{C_i + 3\tau_{bit}}{T_i} \right) \leq 1$.

We rewrite Equation 1 and Equation 2 into a single expression since our probabilistic equations, in the following section, will be based on having such an expression. Having a single expression we will be able to separate the "fixed size" part of the calculations from the "varying size part" based on distributions. The new expression is

$$\begin{aligned} R_i^n &= J_i + B_i + C_i \\ &+ \sum_{j \in hp(i)} I_j \left(R_i^{n-1} - J_i - C_i \right) (C_j + 3\tau_{bit}) \end{aligned} \tag{3}$$

where $I_j(t)$ is defined as the worst-case number of periodic message releases, for a message j, in a time interval of t

$$I_j(t) = \left\lceil \frac{t + J_j + \tau_{bit}}{T_j} \right\rceil \tag{4}$$

where J_j is the worst-case release jitter, and T_j is the period of the message.

As Equation 2, Equation 3 is a recurrence relation. The only difference is that the first approximation is in this case set to $R_i^0 = J_i + C_i$.

2.2. The Bit-Stuffing Mechanism

In CAN, six consecutive bits of the same polarity (111111 or 000000) is used for error and protocol control signalling. To avoid these special bit patterns in transmitted frames, a bit of opposite polarity is inserted after five consecutive bits of the same polarity. By reversing the procedure, these bits are then removed at the receiver side. This technique, which is called *bit-stuffing*, implies that the actual number of transmitted bits may be larger than the size of the original frame, corresponding to an additional transmission delay which needs to be considered in the analysis.

Let us first define the number of bits, beside the data part in the frame, which are exposed to the bit-stuffing mechanism as $g \in \{34, 54\}$. This since we have either 34 (CAN standard format) or 54 (CAN extended format) bits (beside the data part in the frame) which are exposed to the bit-stuffing mechanism. 10 bits in the CAN frame are **not** exposed to the bit-stuffing mechanism (see Figure 2). Now let us define the number of bytes of data in CAN message frame i as $L_i \in [0, 8]$. Recall, a CAN message frame can contain 0 to 8 bytes of data. According to the CAN standard [10], the total number of bits in a CAN frame before bit-stuffing is therefore

$$8L_i + g + 10 \tag{5}$$

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

$$8L_i + g + 10 + \left\lfloor \frac{g + 8L_i - 1}{4} \right\rfloor \tag{6}$$

Intuitively the above formula captures the number of stuffed bits in the worst case scenario, shown in Figure 3.

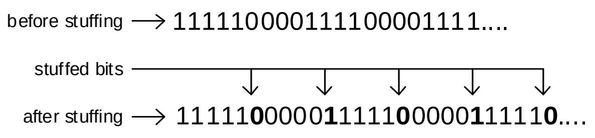

Figure 3. The worst case scenario when stuffing bits.

Let τ_{bit} be the worst-case time taken to transmit a bit on the bus – the so-called *bit time*. The worst-case time taken to transmit a given frame i is therefore

$$C_i = \left(8L_i + g + 10 + \left\lfloor \frac{g + 8L_i - 1}{4} \right\rfloor\right) \tau_{bit} \tag{7}$$

3. New Approach

The expression (6) describes the length of a CAN frame in the worst case. However, in our previous work [8, 9] we represent the number of stuff-bits as a distribution. By using a distribution of stuff-bits instead of the worst-case number of stuff-bits, we obtain a distribution of response-times allowing us to calculate less pessimistic (compared to traditional worst-case) response-times based on probability.

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

From [9] we can extract 9 different distributions of stuff-bits depending on the number of bytes of data in the CAN message frame. We define Υ_{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 $n = \Upsilon(p)$ as the worst-case number of stuff bits, n, to expect with a probability p based on the stuff-bit distribution Υ, i.e., $\sum_{x=n+1}^{\infty} P(x) \leq p$, or to express it in another way, the probability of finding more than n stuff bits, based on the stuff-bit distribution Υ, 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 [9]) that CAN message frames are independent in the sense of number of stuff-bits, we can define $\prod_n \Upsilon$ as the joint distribution corresponding to the combination of n distributions of stuff-bits, i.e., the number of stuff-bits caused by a sequence of n messages sent on the bus is described by $\prod_n \Upsilon = \underbrace{\Upsilon \times \Upsilon \times \cdots \times \Upsilon}_{n}$, where $\times$ denotes multiplicative combination of discrete distributions, as illustrated in the example below. If the distributions happens to be equal, $\overline{\prod_n} \Upsilon$ is defined as the joint distribution of n *equal* distributions of stuff-bits, i.e., the number of data bytes are the same for all messages considered by the expression.

3.1. Example

As an illustration, let us use an example where we assume $\Upsilon = \{(0, 0.1), (1, 0.8), (2, 0.1)\}$. Calculating $\overline{\prod_2} \Upsilon$ is done by multiplying the probabilities for all combinations of stuff-bits, i.e., $(a, P(a)) * (b, P(b)) = (a+b, P(a)*P(b))$ where $a, b \in \{0, 1, 2\}$. The result of a multiplication is a new number of stuff-bits with a corresponding probability. In our example the multiplication yields

$$\begin{aligned}\overline{\prod_2} \Upsilon =& \{(0, 0.01), (1, 0.08), (2, 0.01), (1, 0.08), (2, 0.64),\\ & (3, 0.08), (2, 0.01), (3, 0.08), (4, 0.01)\}\end{aligned} \tag{8}$$

However, all probabilities in $\overline{\prod_2} \Upsilon$ of equal number of stuff-bits are added together leaving

$$\overline{\prod_2} \Upsilon = \{(0, 0.01), (1, 0.16), (2, 0.66), (3, 0.16), (4, 0.01)\} \tag{9}$$

In our example, with $p = 10^{-1}$, $\Upsilon(p) = 1$ and $\left(\overline{\prod_2} \Upsilon\right)(p) = 3$.

3.2. Probabilistic Worst-Case Response-Time

In order to include the bit-stuffing distributions in Equation 3 we need to redefine C_i and B_i to $C_i(p)$ and $B_i(p)$ where

- $C_i(p)$ is the transmission time of message i

  $$C_i(p) = c_i + \Upsilon_{L_i}(p)\tau_{bit} \qquad (10)$$

  where Υ_{L_i} is the distribution of stuff-bits in the message, and c_i is the transmission time of message i excluding stuff-bits

  $$c_i = (8L_i + g + 10)\,\tau_{bit} \qquad (11)$$

  where 10 is the number of bits in the CAN frame **not** exposed to the bit-stuffing mechanism.

- $B_i(p)$ is the blocking time caused by message i having to wait for a lower priority message sent on the bus. Since the bus is non pre-emptive, the worst-case scenario is that the biggest (in size) lower priority message just started its transmission when message i becomes ready to transmit. Thus we can define the blocking time of a message i as

  $$B_i(p) = b_i + \Upsilon_{\max\limits_{k\in lp(i)}(L_k)}(p)\tau_{bit} \qquad (12)$$

  where $\Upsilon_{\max\limits_{k\in lp(i)}(L_k)}$ is the distribution of stuff-bits of the blocking message k (the biggest lower priority message), and b_i is the blocking time not considering the bit-stuffing mechanism

  $$b_i = \max_{k\in lp(i)}(c_k) + 3\tau_{bit} \qquad (13)$$

  where $3\tau_{bit}$ is the inter-frame space. Note that (12) is pessimistic in the sense that we always assume that we will be blocked by a message. Taking probability of blocking actually occurring into consideration as well as not always assuming biggest blocking message would give a less pessimistic result. However, since we are basing the analysis on a “critical instant”, we create a worst-case scenario but we use distributions of values instead of worst-case ones when calculating the response-time.

Taking the probabilistic definitions of Equation 10 and Equation 12 into consideration we can reformulate Equation 3 as

$$\begin{aligned} {R_i}^n(p) &= J_i + b_i + c_i \\ &\quad + \sum_{j\in hp(i)} I_j({R_i}^{n-1}(p) - J_i - c_i)\,(c_j + 3\tau_{bit}) \\ &\quad + \Psi_i(p)\tau_{bit} \end{aligned} \qquad (14)$$

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

$$\Psi_i = \Upsilon_{\max\limits_{k\in lp(i)}(L_k)} \times \Upsilon_{L_i} \times \prod_{j\in hp(i)} \overline{\prod_{I_j(R_i(p)-J_i-c_i)}} \Upsilon_{L_j} \qquad (15)$$

where $\Upsilon_{\max\limits_{k\in lp(i)}(L_k)}$ is the distribution of stuff-bits caused by the longest lower priority blocking message, Υ_{L_i} is the distribution of stuff-bits in the message under analysis, and $\prod_{j\in hp(i)} \overline{\prod_{I_j(R_i(p)-J_i-c_i)}} \Upsilon_{L_j}$ is the distribution of stuff-bits in all interfering messages of higher priority sent before message i will be sent, i.e., the higher priority messages sent causing message i to be queued.

Having the distribution Ψ_i, a proper total number of stuff-bits is selected depending on the desired probability of response time violation p, i.e., for every step in the recurrence relation (14), a value $\Psi_i(p)$ must be extracted from Equation 15.

3.3. Complexity

Regarding the complexity of the analysis, the dominating component is the calculation in Equation 15. Since all parameters in Equation 15 are distributions, and distributions are multiplied together causing multiplications of all combinations of stuff-bits, the complexity of solving the expression is as follows

$$\mathcal{O}(k^l) \qquad (16)$$

where l is the number of messages involved in Equation 15, and k is the number of stuff-bits in the biggest size message, having largest number of stuff-bits in its distribution. However, due to the iterative nature of the equations, solving Equation 15 can be done with a much lower complexity. In fact, the complexity of calculating the joint distribution can be reduced to

$$\mathcal{O}(l * k^2) \qquad (17)$$

since we in each iteration can reduce the number of considered values to $k * l$ by adding all values with equal number of stuff-bits together, as illustrated in Section 3.1.

Priority (ID)	Bytes	C_i (ms)	T_i (ms)	D_i (ms)	R_i (ms)	$p =$ $R_i(p)$ (ms)	10^{-24} gain (%)	$p =$ $R_i(p)$ (ms)	10^{-12} gain (%)	R_i^{sim} (ms)
17	1	0.480	1000	5	1.416	1.384	2.26	1.328	6.21	0.680
16	2	0.560	5	5	2.016	1.936	3.97	1.864	7.54	1.240
15	1	0.480	5	5	2.536	2.448	3.47	2.360	6.94	1.720
14	2	0.560	5	5	3.136	3.032	3.32	2.920	6.89	2.280
13	1	0.480	5	5	3.656	3.536	3.28	3.424	6.35	2.760
12	2	0.560	5	5	4.256	4.120	3.20	4.000	6.02	3.320
11	6	0.864	10	10	5.016	4.840	3.51	4.720	5.90	4.184
10	1	0.480	10	10	8.376	5.368	35.91	5.248	37.34	4.664
9	2	0.560	10	10	8.976	8.480	5.53	8.336	7.13	5.224
8	2	0.560	10	10	9.576	9.144	4.51	9.000	6.02	8.424
7	1	0.480	100	100	10.096	9.728	3.65	9.592	4.99	8.904
6	4	0.712	100	100	19.096	15.256	20.11	10.304	46.04	9.616
5	1	0.480	100	100	19.616	18.472	5.83	18.176	7.34	10.096
4	1	0.480	100	100	20.136	19.224	4.53	18.968	5.80	18.320
3	3	0.632	1000	1000	28.976	19.928	31.23	19.704	32.00	18.952
2	1	0.480	1000	1000	29.496	27.920	5.34	20.400	30.84	19.432
1	1	0.480	1000	1000	29.520	28.352	3.96	27.944	5.34	19.912

Table 1. SAE CAN messages.

3.4. Example

To illustrate our method we use a small example with 3 messages, message 1-3, where message 1 has the highest priority, and message 3 the lowest priority. We assume that we have no jitter, i.e., $J = 0$ for all messages, and that all messages have the same size. Note that the assumption regarding the message length to be equal is just for simplicity for the reader. This is not a requirement. We assume $c = 10$, $\tau_{bit} = 1$, and Υ is as in Section 3.1. Finally, again for simplicity, all message periods are so big causing Equation 4 never to exceed 1, i.e., $I(t) = 1$.

Based on our assumptions, the worst-case scenario for message 2 would be as illustrated in Figure 4, i.e., message 2 is blocked by message 3 (the lowest priority message) and delayed by message 1 (the highest priority message).

Using Equation 14 we can calculate the response time $R_2(p)$ as

$$R_2(p) = b_2 + c_2 + (c_1 + 3\tau_{bit}) + \Psi_2(p)\tau_{bit} \qquad (18)$$

where $b_2 = c_3 + 3\tau_{bit}$ and $\Psi_2 = \Upsilon_{L_3} \times \Upsilon_{L_2} \times \Upsilon_{L_1} = \bar{\prod_3}\Upsilon$ (since $L_1 = L_2 = L_3$) where $\bar{\prod_3}\Upsilon$ is calculated to be

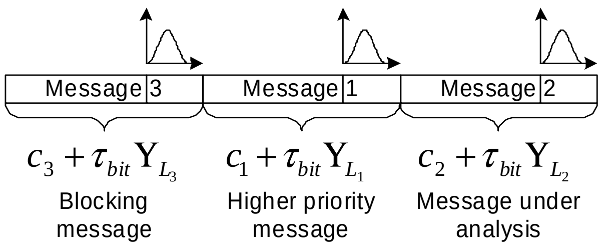
Message 3
Message 1
Message 2
$c_3 + \tau_{bit}\Upsilon_{L_3}$
$c_1 + \tau_{bit}\Upsilon_{L_1}$
$c_2 + \tau_{bit}\Upsilon_{L_2}$
Blocking message
Higher priority message
Message under analysis

Figure 4. Worst-case message sequence for message 2.

$$\bar{\prod_3}\Upsilon = \{(0, 0.001), (1, 0.024), (2, 0.195), (3, 0.56), (4, 0.195), (5, 0.024), (6, 0.001)\} \qquad (19)$$

We select an acceptable probability of worst-case response time violation p to be 10^{-1}. Based on p, $\Psi_2(p) = 4$, causing $R_2(p) = (10 + 3) + 10 + (10 + 3) + 4 = 40$.

4. Evaluation

In order to demonstrate the performance of our new approach for calculating a probabilistic worst-case response

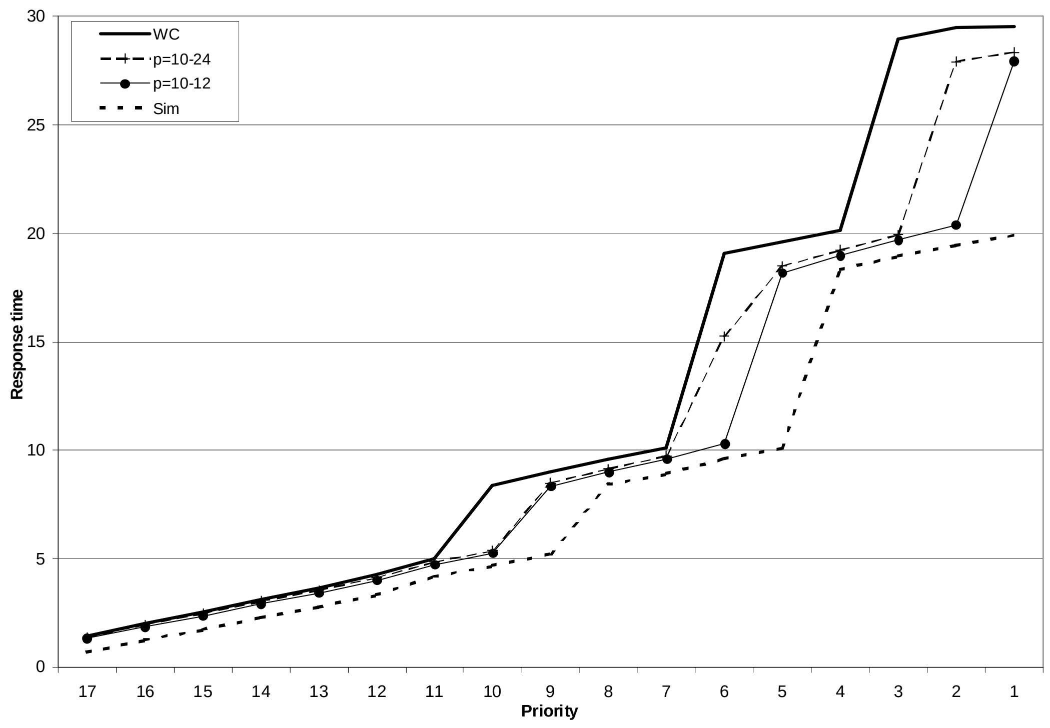

Figure 5. Message response times (priority is the message ID as in Table 1).

time we make use of the widely published simplification [14] of the Society of Automotive Engineers (SAE) benchmark [11].

We use a bus speed of 125kbit/s, and we select the acceptable probability of violating the calculated worst case response time, p, to be 10^{-24} and 10^{-12} respectively. Then, we calculate the worst-case response time both according to the traditional approach (1) and the probabilistic approach (14). The response times of all messages of the subset of SAE messages are shown in Table 1, where R_i denotes the results of traditional analysis and $R_i(p)$ the results of our new probabilistic analysis. To have some "real" response times to compare the analytic ones with, we simulated the SAE message set using the worst-case transmission times. The system was simulated for 2000000 ms. The worst-case measured response time is presented as R_i^{sim} in the rightmost column of Table 1. Note that the difference between the simulated value and the analytic worst-case is due to that in the simulation all messages are released at time "0". Hence, the worst-case blocking as defined by Equation 12 might not occur due to the phasings of messages.

What we see in Table 1 is that the probabilistic response times $R_i(p)$ are significantly lower than the traditional worst-case response times R_i. An interesting observation is that the gain is substantially higher for some messages. The reason for this is that a slight additional interference, e.g., caused by an additional stuff-bit, will in these cases extend the response-time such that transmission will be delayed by one or more additional higher priority message transmissions. Note that all calculated probabilistic response times are never optimistic in comparison with the simulation result (as seen in Figure 5). This even though we are using worst-case transmission times. Using bit-stuffing distributions in the simulation would give even shorter response times.

5. Conclusions

In this paper we have presented a new probabilistic approach to calculate response times for messages in the Controller Area Network. The key element to this approach is that we use bit-stuffing distributions instead of worst-case values. The performance of our method is evaluated using a subset of the SAE benchmark.

Our main motivation for calculating probabilistic response-times is that they allow us to reason about tradeoffs between reliability and timeliness. We have in [6] presented a method for such analysis of controller area networks subject to external interference. An obvious next step would be to integrate the bit-stuffing distribution based analysis presented here with that analysis.

Acknowledgements

The authors wish to express their gratitude to the anonymous reviewers for their helpful comments. The work presented in this paper was supported by the Swedish Foundation for Strategic Research (SSF) via the research programme ARTES, the Swedish Foundation for Knowledge and Competence Development (KK-stiftelsen), and Mälardalen University.

References

[1] A. Atlas and A. Bestavros. Statistical Rate Monotonic Scheduling. *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 123–132, December 1998.

[2] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[3] I. Broster and A. Burns. Timely Use of the CAN Protocol in Critical Hard Real-Time Systems With Faults. *Proceedings of the* 13^{th} *Euromicro Conference on Real-Time Systems*, June 2001.

[4] A. Burns. Preemptive Priority Based Scheduling: An Appropriate Engineering Approach. Technical Report YCS 214, University of York, 1993.

[5] CAN Specification 2.0, Part-A and Part-B. CAN in Automation (CiA), Am Weichselgarten 26, D-91058 Erlangen. http://www.can-cia.de/, 2002.

[6] H. Hansson, T. Nolte, C. Norström, and S. Punnekkat. Integrating Reliability and Timing Analysis of CAN-based Systems. *IEEE Transaction on Industrial Electronics*, 49(6), December 2002.

[7] S. Manolache. Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times. *Licentiate Thesis No. 985, Dept. of Computer and Information Science, IDA, Linköping University, Sweden*, December 2002.

[8] T. Nolte, H. Hansson, and C. Norström. Using Bit-Stuffing Distributions in CAN Analysis. *IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01)*, December 2001.

[9] T. Nolte, H. Hansson, and C. Norström. Minimizing CAN Response-Time Analysis Jitter by Message Manipulation. *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, pages 197–206, September 2002.

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

[11] SAE. Class C Application Requirement Considerations-SAE J2056/1. *SAE Handbook*, pages 23.366–23.371, June 1993.

[12] L. Sha, R. Rajkumar, and J. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[13] K. W. Tindell and A. Burns. Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks. Technical Report YCS229, Dept. of Computer Science, University of York, June 1994.

[14] K. W. Tindell, A. Burns, and A. J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[15] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). *Proceedings of RTSS'94* - 15^{th} *IEEE Real-Time Systems Symposium*, pages 259–265, December 1994.

[16] J. Xu and D. L. Parnas. Priority Scheduling Versus Pre-Run-Time Scheduling. *Real-Time Systems Journal*, 18(1):7–23, January 2000.