Probabilistic Analysis of CAN with Faults

Ian Broster, Alan Burns
University of York
UK
{ianb, burns}@cs.york.ac.uk

Guillermo Rodríguez-Navas
Universitat de les Illes Balears
Spain
vdmigrg0@clust.uib.es

Abstract

As CAN (Controller Area Network) is increasingly used in safety-critical applications, there is a need for accurate predictions of failure probability. In this paper we provide a general probabilistic schedulability analysis technique which is applied specifically to CAN to determine the effect of random network faults on the response times of messages. The resultant probability distribution of response times can be used to provide probabilistic guarantees of real-time behaviour in the presence of faults. The analysis is designed to have as little pessimism as possible but never be optimistic. Through simulations, this is shown to be the case. It is easy to apply and can provide useful evidence for justification of an event-triggered bus in a critical system.

1 Introduction

To be able to argue about the dependability of a system, one must be able to determine the dependability of its components. In a distributed system, determining the dependability of the communication channel is of particular importance, especially when this component is susceptible to electromagnetic interference (EMI). Therefore in safety-critical systems it is vital to define a mechanism able to determine how the system dependability is affected by faults on communication.

This paper deals with a specific bus: CAN [3, 8] because of its increasing use in safety-critical systems. The error detection and recovery features of CAN provide a high tolerance to network faults. However, as the recovery mechanism of CAN involves retransmitting any corrupted messages, network faults cause extra overhead and can delay messages on the bus, so analysis techniques to deal with the effect of faults are required.

It is standard practice to assume an upper bound on the effect of faults, this allows worst case response time analysis to be performed [18]. However, network faults are difficult to predict because there is often no way to know what external influences the system will be exposed to during its lifetime. It is more useful to move to a probabilistic scheme where events such as faults are modelled using probabilities. Design decisions can then be made based on probabilities and consequences of failure. In order to do this, analysis is required that can deal with probabilistic fault models without excessive pessimism.

This paper introduces an analysis technique that can produce a distribution of worst case response times when there are probabilistic events in the system. It is applied specifically to CAN [3, 8] to determine the reliability of the bus with respect to deadline failures in the presence of random network faults (modelled as a Poisson distribution).

The motivation for this research is to determine whether and how an event-triggered bus such as CAN may be used in a critical system with faults. The analysis presented in section 3 allows the probability of timing failures to be determined, and a case study in section 4 is used to illustrate the analysis. On the basis of this and other research, we believe that the flexibility of an event-triggered bus (such as CAN) can be *exploited* to provide fault tolerance and produce highly reliable and predictable real-time systems.

2 Previous work

CAN's successful history in the automotive industry as a flexible embedded bus has driven its use in other applications such as factory-control networking, elevator control systems, machine tools, photo-copiers and many more. The advantages of using a bus such as CAN may include: reduced design time (readily available components and tools) and lower connection costs (lighter, smaller cables and connectors) [17].

However for use in safety-critical avionics applications, CAN is still not considered to be sufficiently reliable and predictable. In such systems, guaranteeing the performance of the bus is crucial to certification. Nevertheless there is mounting evidence that CAN is able to support safety-critical avionics applications, many papers have investigated possible failures of CAN, shown its weaknesses and proposed solutions to those weaknesses [6, 9, 12, 19]. In this paper we are concerned with one particular aspect of CAN in critical systems: the effects of network faults on the real-time properties of the bus. It will be assumed that other possible weaknesses of CAN such as the “last but one bit” problem [12, 13] and error-passive mode problem [15] are otherwise solved.

A method to analyse worst case real-time behaviour of a CAN bus was produced by Tindell and Burns [18]. They showed by applying processor scheduling analysis to the CAN bus, 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 retransmitting 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 [18] to show how the schedulability analysis is performed in the presence of errors in the channel. The model is based on a minimum inter-arrival 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 [14] extends the work of Tindell and Burns 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 inter-arrival time. Except for the more general fault model, the rest of the schedulability analysis is performed like [18].

Both Tindell’s and Punnekkat’s analyses are useful because they illustrate how to perform schedulability analysis of CAN under fault conditions. However, they use models based on a minimum inter-arrival 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 following a Poisson distribution [5]. 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 inter-arrival time.

Unlike Tindell and Punnekkat, Navet [10] proposes a probabilistic fault model, which incorporates the uncertainly of faults caused by EMI. The fault model suggested by Navet is modelled as a stochastic process which considers both the frequency of the faults and their gravity. In that model, faults in the channel occur following a Poisson law and can be either single-bit faults (which have a duration of one bit) 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 modelled 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 non-zero; therefore every system is inherently unschedulable. Hence Navet’s analysis does not try to determine whether a system is schedulable (as [18, 14]), 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. The method provided by Navet for estimating the WCDFP of a message is now briefly presented.

As shown in [18], a message can be guaranteed to meet its deadline even if it has to be retransmitted because of errors in the channel. Nevertheless, as the response time of a message increases with the number of retransmissions, meeting its deadline can only be guaranteed up to a given number of retransmissions. Therefore, determining such a number of tolerated retransmissions is the first step to calculate the probability of missing a deadline.

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_m errors occur during $R_{m_{max}}$. This probability can be analytically calculated as the fault model assumed by Navet is a generalised 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 are 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 [10], 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 Navet’s analysis, a burst error of duration u bits is treated as a sequence of u single bit faults [10, eqn. (7)], each causing a maximal error overhead (an error frame and the retransmission of a frame of higher or equal priority). This assumption allows consideration of the fault model as a generalised Poisson process and facilitates solving the WCDFP equation. However, this assumption is inconsistent with the CAN protocol specification [3, 8] since in reality a burst error can only cause retransmission of 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 [7]. This work points out that errors happening during bus idle do not cause any message retransmission, and therefore those errors cause an 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 which is a simplification of the fault model presented in [14]. 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.

We believe that modelling arrivals of errors with a random distribution, as done in [10], allows a more generalised solution. In this paper we propose an analysis that provides an accurate probability of deadline failure without excess pessimism, based on the assumption that faults are randomly distributed.

3 Analysis

In this section, an analysis is developed which provides a probability distribution of worst case response times under a random arrival fault model. This in turn may be used to determine an accurate probability of deadline failure.

3.1 Pessimistic Approach

In the context of this work (critical real-time), an analysis should never be optimistic — there must be no opportunity for the analysis to give a lower response time or higher cumulative probability than the system it is modelling. Therefore, in each stage of the analysis, there must be no optimistic assumptions. The aim is to have a pessimistic solution, but with as little pessimism as possible.

Therefore, the analysis presented in this section starts with the strict worst case analysis and removes sources of pessimism where possible. There are several areas in the analysis where a worst case is still assumed. It would be inappropriate to assume for example a mean or other expected value for any parameter, otherwise it would risk being optimistic. In particular, the analysis always considers:

- the first invocation after a critical instant where there is the maximum possible queueing delay due to other frames/tasks, other invocations in the hyperperiod are not considered;
- that faults always have the maximum possible impact — most of the time this will not occur, but we show later that it does not lead to an excessive level of pessimism.

3.2 Response Time

The analysis considers the critical instant where all higher priority messages are simultaneously queued (imposing maximum interference) and the longest lower priority message has just started (imposing maximum blocking). This is the known worst case scenario.

We may begin with the usual worst case response time equations for CAN [18]. The equations here are slightly modified [4] by separating the inter-frame space from the data frame. This removes a small source of pessimism by considering that the frame is available as soon as the last bit has been received, rather than only after the inter-frame space that follows it has finished.

$$R_i = B_i + C_i + I_i(R_i) + E_i(R_i) \qquad (1)$$

Where

- C_i is the worst case transmission time [14, 11] (the time it takes, in the worst case to send frame i) assuming no errors, maximum bit stuffing and not including the inter-frame space that follows the frame:

$$C_i = \left(44 + 8b + \left\lfloor \frac{34 + 8b - 1}{4} \right\rfloor \right) * \tau \qquad (2)$$

  where τ is one bit-time and b is the number of data bytes in the frame (0 to 8).

- B_i is the worst case blocking time, this is the maximum time a message may need to wait due to a lower priority message on the bus:

$$B_i = \max_{\forall k \in \mathsf{lp}(i)} (C_k) + S \qquad (3)$$

  where S is the inter-frame space (3 bit-times) and $\mathsf{lp}(i)$ is the set of messages with lower priority than i:

- $I_i(t)$ is the worst case interference that message i may receive in t time units:

$$I_i(t) = \sum_{j \in \mathsf{hp}(i)} \left\lceil \frac{t - C_i + J_j + \tau}{T_j} \right\rceil (C_j + S) \qquad (4)$$

  where $\mathsf{hp}(i)$ is the set of messages with higher priority than i and
  J_j is the worst case release jitter of frame j. The τ is

used to eliminate 'edge effects' where a high priority frame becomes ready as a medium priority one completes [2].

- $E_i(t)$ is some function which returns the overhead due to faults in period t, this will be explained in section 3.3.

Equation (1) may be solved iteratively

$$t_i^{n+1} = B_i + C_i + I_i(t_i^n) + E_i(t_i^n) \tag{5}$$

with $t_i^0 = C_i$. Iteration terminates when $t_i^{n+1} = t_i^n$ provided $t_i^n \leqslant T_i - J_i$. The final value of t_i is the worst case response time of frame i:

$$R_i = t_i^n + J_i \tag{6}$$

The above equations are developed in the following sections.

3.3 Fault Model

We shall consider a simple fault model and define the error overhead function $E_i(t)$ accordingly.

The simple fault model used is that faults arrive randomly, with a Poisson distribution $F \sim \mathcal{P}o(\lambda)$. Therefore, the probability of exactly m faults occurring in any time interval t is:

$$p_t(F = m) = \frac{e^{-\lambda t}(\lambda t)^m}{m!} \tag{7}$$

To avoid optimism, it is assumed that each fault causes the maximum length error frame (E_{max}) and occurs on the last bit of the longest frame, such that the additional overhead due to one fault is equal to one error frame and one retransmission of a higher priority message:

$$M_i = E_{max} + \max_{j \in \mathsf{hep}(i)} C_j \tag{8}$$

where $\mathsf{hep}(i)$ is the set of messages with higher or equal priority to i.[1]

Therefore the error overhead function, $E_i(t)$, is a random distribution:

$$E_i(t) = \begin{cases} 0 & \text{with probability } p_t(F=0) \\ M_i & \text{with probability } p_t(F=1) \\ 2M_i & \text{with probability } p_t(F=2) \\ 3M_i & \text{with probability } p_t(F=3) \\ \cdots & \cdots \end{cases} \tag{9}$$

or more generally:

$$E_i(t) = mM_i \text{ with probability } p_t(F = m) \tag{10}$$

[1]Note that even though a fault can occur in a lower priority frame, a frame with lower priority than i cannot be retransmitted while frame i is queued for transmission, so we use $\mathsf{hep}(i)$.

It should be noted that this error function is always pessimistic — $E_i(t)$ is always at least as great as would occur in a real bus.

There are an infinite number of values for $E_i(t)$ since $m \in 0..\infty$. However, we can neglect values of m where $p_t(F = m) < \epsilon$, where ϵ is some suitably small probability that is so small that designers consider the risk acceptable, for example $\epsilon = 10^{-18}$.

Therefore we may enumerate all significant values of m and consider each case individually, with its associated probability.

3.4 Search Space

When equation (5) is solved iteratively, it generates a monotonically increasing set of values of t_i and therefore a set of non-overlapping intervals exists: $\{(0, t_i^0],\ (t_i^0, t_i^1],\ (t_i^1, t_i^2],\ \cdots\ (t_i^k, t_i^k]\}$. Within each interval, the error overhead function, $E_i(t)$, is evaluated for each significant value of $E_i(t_i^n - t_i^{n-1})$.

In order to do this, a probability tree is used. An example is shown in Figure 1 which is used in the following walkthrough.

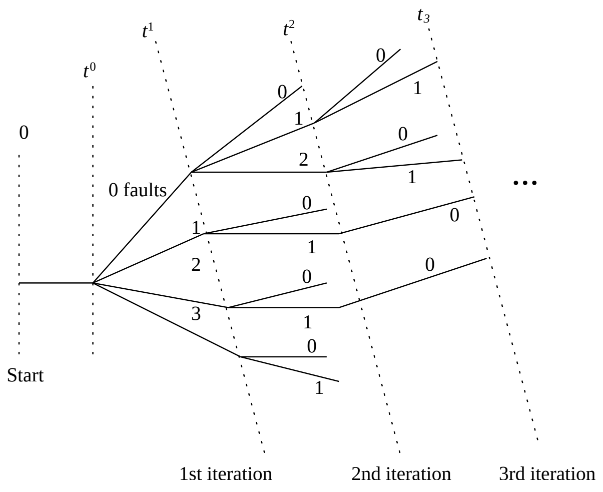

Figure 1. Example Probability Tree for Simple Fault Model

Using equation (5) and the initial value $t_i^0 = C_i$ we consider the set of all possible numbers of faults in the interval $(0, t_i^0]$ for which there is a significant probability.

$$A_{t_i^0} = \{f | p_{(t_i^0 - 0)}(F = f) \geqslant \epsilon\} \tag{11}$$

Each member of $A_{t_i^0}$ is used to derive a set of possible values for t_i^1 with an associated probability $p(t_i^1) = p_{(t_i^0 - 0)}(F = f)$. In the example there are four significant possibilities: in the interval there could be 0, 1, 2 or 3 faults, (the probability of more than 3 faults is less than ϵ) so we derive four branches of the tree with 4 different values of t_i^1 and associated probabilities.

Following the first branch (0 faults), we next consider the interval $(t^0, t^1]$ and apply the Poisson equation (7) to the interval $(t^0, t^1]$. (We may do this because the probability of any number of faults in an interval is independent from the history of faults before the start of the interval.) Therefore we apply equation (7) to determine that there are three *significant* possibilities: there are 0, 1 or 2 faults in the interval $(t_0, t_1]$. So the three branches provide three possibilities for t_i^2 and associated probabilities of $p(t_i^2) = p(t_i^1) \cdot p_{(t_i^1 - t_i^0)}(F = f)$.

We continue to recursively explore the tree. For subsequent iterations of equation (5), the error overhead function must be applied with care. At each branch in the tree, the path to the root of the tree has been fixed, therefore when $E_i(t)$ is evaluated, it must only consider the time between t_i^n and t_i^{n-1}. Yet it must not neglect the previous error overhead. We may write:

$$t_i^{n+1} = B_i + C_i + I_i(t_i^n) + \sum_{j=0}^{n-1} E_i(t_i^{j+1} - t_i^j) \qquad (12)$$

However, equation (12) is easier to understand and implement if written as:

$$t_i^{n+1} = B_i + C_i + I_i(t_i^n) + E_i^n \qquad (13)$$

and E_i^n is the total error overhead for the previous iterations in the path to the root of the tree, plus the overhead added this iteration:

$$E_i^n = E_i^{n-1} + E_i(\Delta t) \qquad (14)$$

where $\Delta t = t_i^n - t_i^{n-1}$

Evaluation continues until all significant branches are explored. The base cases for recursion are:

- $t_i^{n+1} = t_i^n$. This occurs when we consider the probability of finding 0 faults and there is no further interference to consider. In this case, t_i^n represents a possible value for R_i with a known probability. The pair $\langle t_i^n, p(t_i^n)\rangle$ is recorded.

- $p(t_i^n) < \epsilon$. A path is so unlikely to occur that it is insignificant, in which case we may ignore this path.

- $t_i^{n+1} > T_i - J_i$. Any analysis based on equation (5) cannot handle the possibility that the response time is greater than the period of a message. Therefore, when this occurs, we must record the probability that the message set is unschedulable.

3.5 Interpretation of Distribution

The immediate result of the analysis is a set of pairs $R_i = \langle t_i, p(t_i)\rangle$. More usefully, a cumulative probability distribution can be plotted.

The nature of the analysis means that we can expect some very small probabilities down to ϵ, as well as larger values; we are interested in all values. On a linear scale we would not expect to see the smaller probabilities, so a logarithmic scale is appropriate (indeed, on a linear scale, a graph shows very little of interest). However, in order to plot a cumulative distribution, the graph must display $(1 - cumulative\,probability)$ to ensure that the small changes in probability are close to 0 rather than close to 1. The values on this axis can be interpreted as an upper bound on the probability of the response time exceeding the corresponding value on the horizontal axis.

Unfortunately, these transformations can be difficult to visualise, leaving a somewhat counter-intuitive plot. So by means of an illustrative example in Figure 2, a typical output of the analysis is shown in the same form as it is in the rest of the paper.

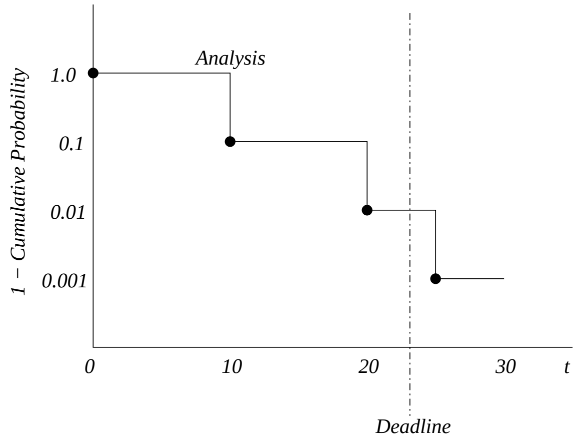

Figure 2. Interpretation of Results

Care must be taken when interpreting the graph as it represents discrete data, not a continuous function. The correct way to join the points is as shown in Figure 2 so that the resulting line represents an upper bound on the results.

With reference to the graph, one can infer for example that:

- $p(t \geqslant 10) \leqslant 0.1$
  The analysis records that $p(t \geqslant 10) = 0.1$, but the analysis is pessemistic, therefore the probability of a real frame having a response time longer than 10 is less than or equal to 0.1.

- $p(t \geqslant 19) \leqslant 0.1$

- $p(t < 10) > 0.9$

- the probability of a given message being delayed beyond the deadline is less than or equal to 0.01. This is equivalent to Navet's worst case deadline failure probability (WCDFP).

- if data obtained from extended monitoring of a real system or by simulation is plotted on the same axes, then if the analysis is pessimistic, the real data should always appear further to the left of the graph than the analysis line.

3.6 Probabilities and ϵ

The algorithm necessarily cuts improbable branches of the tree. However, if there are a large number of branches that have probabilities of just below ϵ then their summed probability can be significant. Therefore to assess the *completeness* of the search, the *coverage* can be defined as:

$$Q_i = p(N) + \sum_{\forall \langle t,p \rangle \in R_i} p \tag{15}$$

where $p(N)$ is the probability of being unschedulable, recorded by the analysis when $t_i^{n+1} > T_i - J_i$.

To be safe, any branches that are not covered should be treated as unschedulable. Therefore, the WCDFP can be found directly from the probability of being less than the deadline, but taking into account any paths that are not covered:

$$p(R_i > D_i) = 1 - \left(\sum_{\forall \langle t,p \rangle \in R_i | t \leqslant D_i} p \right) + (1 - Q_i) \tag{16}$$

The parameter ϵ is important to the analysis because it controls the amount of coverage achieved. However, increased coverage comes at the cost of execution time because as the coverage increases, the number of branches increases exponentially. Values of ϵ between 10^{-10} and 10^{-20} are useful, depending on the message set and system requirements. A value of ϵ too small will result in infeasible execution times for the algorithm, the use of larger values will lose coverage of the search space and therefore lose accuracy for low probabilities.

Setting ϵ can be done on a message by message basis, by considering the failure requirements as follows. The resultant probability distribution refers to one single invocation of the message. It is common for a dependable system to have reliability requirements in the form "less than 10^{-9} failures per hour". If we assume that timing failures are independent, then we can calculate the failure probability requirement *per invocation*. For example, for a periodic message with $T = 100$ms, there are 36,000 invocations per hour requiring a probability of failure per invocation of $10^{-9}/36,000 = 2.7 \cdot 10^{-15}$. To ensure precision, a value of ϵ must be chosen such that it is an order of magnitude lower than the probability of failure per invocation. In this case a suitable value is $\epsilon = 10^{-16}$. This value can be adjusted if necessary if the coverage is deemed to be insufficient, or

```
procedure Response($t$, $\Delta t$, $E_{prev}$, $p_{path}$) is
  -- Recursive procedure to solve eqn (13)
  -- $E_{prev}$ is $E_i^{n-1}$ in eqn (14)
  if $\Delta t = 0$ then -- Base case: Converged
    AddToPDF($t$, $p_{path}$);
  else if $t > T_i - J_i$ then -- Base case: only $R \leqslant T$
    AddToPDF(NOTSCHED, $p_{path}$);
  else -- Recursive case
    for $j \in \{n \in \mathbb{N} | p_{path} \cdot$poisson(n, $\Delta t$, lambda)$\geqslant \epsilon\}$ do
      $p_j \leftarrow$poisson(j, $\Delta t$, lambda);
      -- Probability of j errors
      $E_j \leftarrow$Errors(i, j, $\Delta t$);
      -- Overhead of j errors
      $t_{new} \leftarrow C_i + \mathrm{B}(i) + \mathrm{I}(i,t) + E_{prev} + E_j$;
      Response($t_{new}$, $t_{new} - t$, $E_{prev} + E_j$, $p_{path} * p_j$);
    end for;
  end if;
end Response;

for $i \in$ messages do
  ClearPDF();
  Response($C_i, C_i, 0, 1.0$);
  PrintPDF();
end for;
```

Figure 3. Analysis Algorithm

if the algorithm execution time is too long for a particular message set.

3.7 Algorithm, Complexity and Implementation

The analysis can be efficiently implemented as a recursive procedure, exploring the tree depth-first and using a simple data-structure to record the results. An algorithm is shown in Figure 3.

The algorithm is presented in a form designed for clarity. However, it should be noted that the test for the second base case ($pr(path) < \epsilon$) is implicit in the **for** loop. A real implementation of the algorithm should also make a number of further optimisations (such as calculating Interference and Blocking outside the loop). Also, extreme care should be taken if implementing with floating point arithmetic because the algorithm may need to manipulate both large and very small values together while avoiding loss of precision. Functions such as Poisson() are particularly vulnerable.

The tree has the potential to grow exponentially, which would lead to infeasible computation times. However the algorithm keeps execution under control by capping many branches using ϵ. In general, the search tree is very skewed (as in Figure 1), vastly reducing the overall size of the tree. As an example of the complexity, section 4 presents results based on two case studies: an implementation of the analysis on a modern PC explored all the trees of the message sets in a few seconds. The number of branches explored by the analysis ranged from around 500 for the highest priority messages to just over 2,700,000 for the lowest priority message. The maximum recursion depth was 26.

Despite the potential for exponential growth, the analysis is much faster to compute than simulation or monitoring of the bus. This is because simulation would have to be performed for extended periods in order to be likely to observe infrequent scenarios, whereas the analysis systematically detects unlikely scenarios during the computation of the tree.

As noted in section 3.6, the value of ϵ can have a dramatic effect on the computation time because it increases the maximum recursion depth as well as the number of branches at each stage. Nevertheless, infeasible computation times have not been observed for any message set tested so far.

4 Evaluation

In this section, the analysis is evaluated by use of two case studies. The results of the analysis are compared with data obtained through software simulation and to a previous study.

4.1 Message Set

The first case study used was proposed by Navet [10]. The original source was provided by Peugeot-Citroen Automobiles Company. There are 12 periodic messages from six devices in a prototype car. A data-rate of 250kbit/s is used. The message set is not a particularly 'strenuous' one: under no-fault conditions, the worst case response time for all messages is less than the shortest period in the set. Therefore, no frame can experience interference from any other frame more than once. The bus utilisation is only 21.5%. It should be noted that the messages are also schedulable at the slower (and therefore less prone to faults) data-rate of 125kbit/sec, and that the priorities are not ordered rate monotonically. For all messages, the deadline is equal to the period and there is no release jitter. The message set is shown in Table 1 with standard worst case response time calculations shown for comparison.

The analysis in section 3 was applied at the fault rate of $\lambda = 30$ faults per second and $\epsilon = 2.7 \cdot 10^{-15}$. We use this value of λ because it has been frequently used in the past [10, 12] as an expected number of faults in an aggressive environment. Reading the results directly from Table 2 we can get a feel for the analysis. Only two frames are shown, however they are representative of the other frames: all show similar results. It can be seen that the probabilities of messages being significantly delayed are very low. The probability of any message being delayed beyond its deadline is insignificant (less than ϵ, see section 3.6).

The full data set is plotted as a cumulative probability graph in Figure 4. The 'steps' are due to the nature of the error overhead function, $E_i(t)$: it can only return a small number of discrete values, so even though the search tree

Table 1. Example Message Set 1

Priority[2] P_i	Length DLC	Period T_i(ms)	WCRT R_i(ms)
12	8	10	1.028
11	3	14	1.368
10	3	20	1.708
9	2	15	2.008
8	5	20	2.428
7	5	40	2.848
6	4	15	3.228
5	5	50	3.648
4	4	20	4.028
3	7	100	4.448
2	5	50	4.708
1	1	100	4.720

[1]Note that we use here a higher number to indicate a higher priority.

Table 2. Analysis results, (Set 1, $\lambda = 30.0$)

Priority: 12

R_i (ms)	$pr(R_i)$
1.028	0.969631
1.672	0.0293312
2.316	0.000999469
2.960	3.70872e-05
3.604	1.45769e-06
4.248	5.96774e-08
4.892	2.51816e-09
5.536	1.08753e-10
6.180	4.72729e-12
6.824	5.4321e-14
> 10	0

Priority: 5

R_i (ms)	$pr(R_i)$
3.648	0.896336
4.292	0.096218
4.936	0.00698767
5.580	0.000432349
6.224	2.46289e-05
6.868	1.33758e-06
7.512	7.0527e-08
8.156	3.64815e-09
8.800	1.86287e-10
9.444	9.24425e-12
10.088	2.95448e-13
> 50	0

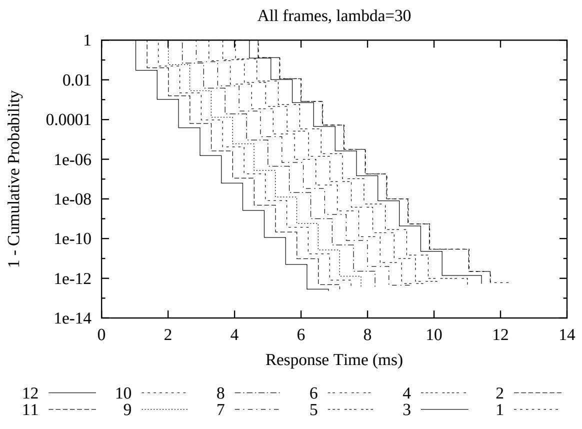

Figure 4. Probability distribution of response times for all frames (Set 1, $\lambda = 30$)

may have many branches, the actual number of branches which have different values of t is very small.

A software simulator has been constructed which accurately models the CAN bus in the presence of faults. It is possible to simulate for extended periods, or to repeatedly simulate conditions after a critical instant.

In order to explore the pessimism and accuracy of the analysis, the analysis results for the frame with priority 5 are compared to the repeated simulation of a critical instant. The same Poisson fault model of $\lambda = 30$ faults per second was used for both the analysis and simulation, and the simulation was run 1,500,000 times. The results of both the simulation and the analysis for this frame are plotted in Figure 5.

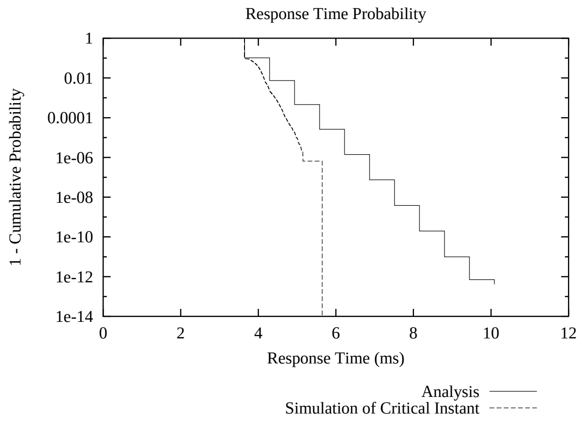

Figure 5. Frame 5, Response Time Probability, Analysis vs. Simulation (Set 1, $\lambda = 30$)

The analysis and the simulation results are very similar. We consider first the response times with probability greater than about 10^{-6}. The analysis is slightly pessimistic: shown by the fact that the analysis line is always to the right of the simulation line. The pessimism here comes mainly from the fact that the analysis always uses a maximal error function $E_i(t)$ whereas the simulation more accurately models network faults occurring in frames. However, the pessimism is not extreme — as can be see from the graph, the lines are very close.

Of particular note are long response times at low priorities as it is these which need to be carefully considered in a critical system. The longer response times suggested in the analysis never occurred in the simulation; they are unlikely to because the simulation run had far less than 10^{14} iterations. We hypothesise that the trend in Figure 5 continues: analysis results are also slightly pessimistic for infrequent events.

The advantage of probabilistic analysis over simulation or monitoring is clear: in order to observe events at probabilities as low as 10^{-14}, we would expect to need to simulate for at least the reciprocal of this number of times. Based on the simulator used here, this would take approximately 30 years of simulation, whereas the analysis took only seconds.

In comparison with Navet [10], the analysis in this paper is far less pessimistic. Navet shows the worst case deadline failure probability for frame 5, $p(R_i > 50ms)$, with $\lambda = 30$ to be approximately 0.045. According to our analysis, the probability of any frame being unschedulable is insignificant. As the deadline was never approached in the 1,500,000 simulation runs, this shows that 0.045 is indeed an overly pessimistic value[3].

4.2 SAE Benchmark

The second case study used to evaluate the analysis is Tindell's widely published simplification [18] of the Society of Automotive Engineers (SAE) benchmark [16]. The messages appear in Table 3. There are 17 periodic data frames with periods ranging from 5ms to 1s, release jitter is neglected. At the bus speed of 125kbit/s, the utilisation is about 85%, which is fairly high and therefore provides a good basis for fault injection. The message set is only just schedulable, frames 12, 9, 8 and 7 are particularly vulnerable to missing their deadlines because their worst case response times are close to their deadlines.

Table 3. SAE CAN message set

Pri	Bytes	C_i	T_i	D_i	R_i
		(ms)	(ms)	(ms)	(ms)
17	1	0.480	1000	5	1.416
16	2	0.560	5	5	2.016
15	1	0.480	5	5	2.536
14	2	0.560	5	5	3.136
13	1	0.480	5	5	3.656
12	2	0.560	5	5	4.256
11	6	0.864	10	10	5.016
10	1	0.480	10	10	8.376
9	2	0.560	10	10	8.976
8	2	0.560	10	10	9.576
7	1	0.480	100	100	10.096
6	4	0.712	100	100	19.096
5	1	0.480	100	100	19.616
4	1	0.480	100	100	20.136
3	3	0.632	1000	1000	28.976
2	1	0.480	1000	1000	29.496
1	1	0.480	1000	1000	29.520

The analysis was performed with $\lambda = 10$ faults per second and $\epsilon = 2.7 \cdot 10^{-15}$. The results of the analysis are similar in form to the previous case study, and are plotted in Figure 6.

[3]The fault model used is similar to Navet's (Poisson, same λ) so it is appropriate to compare these values. Nevertheless, the models *are* different (no burst errors) so a direct comparison should not be misinterpreted.

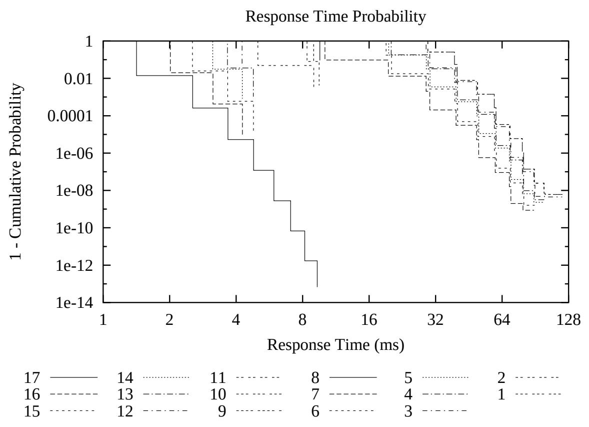

Figure 6. Analysis of all frames (Set 2, $\lambda = 10$)

As the message set is only just schedulable, there is opportunity to explore the probability of deadline failure. The analysis results for message 15 appear in Table 4, expressed to 15 significant figures and are plotted (also with the simulation results) in Figure 7. We find that the coverage from equation (15) is high, $1 - Q_{15} = 1.031 \cdot 10^{-15}$ and the WCDFP for one invocation is then calculated as $1.431 \cdot 10^{-5}$, this being an upper bound for the real probability of frame 15 missing a deadline. Again, the simulations confirm that the analysis is slightly pessimistic in all cases.

Notice that not all the lines in Figure 6 continue down to the very low probabilities. This can be due to a combination of two factors: either the analysis does not consider response times greater than the period of the message; or the coverage is low. The probability at the lowest point on a line can be interpreted as the probability of the message being unschedulable, taking coverage into account.

Examples of both effects appear in this message set: the high and mid-priority frames in Figure 6 (except the very highest priority message) have high coverage but their response times are very close to their periods and so have relatively high probabilities of deadline failure, *e.g.* for message 12, $p(R_{12} > D_{12}) = 0.0416$. Therefore (depending on the particular requirements of the system) the analysis here would indicate that this message set is not acceptable.

For the lowest priority message in this example, the coverage was somewhat lower than would normally be required ($1 - Q_1 = 6.1139 \cdot 10^{-9}$) and so the line stops around this point even though the message is still easily schedulable. This example frame shows the importance of checking the coverage: in this case the coverage can be increased by adjusting ϵ to provide analysis results of longer response times.

Table 4. Message 15 (Set 2, $\lambda = 10$)

R_i (ms)	$pr(R_i)$
0	0
2.536	0.974.958.863.652.502
3.664	0.999,406,490,006,425
4.792	0.999,985,684,829,411
> 5	1.431,517,058,845,04e-05

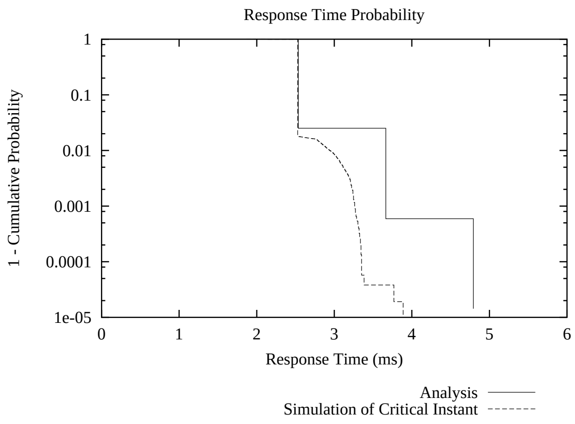

Figure 7. Analysis and Simulation for frame 15 (Set 2, $\lambda = 10$)

5 Conclusion

In this paper a new analysis has been presented which provides a distribution of response times in a system with a random process. It is applied specifically to CAN where the effect of random network faults is considered. The analysis aims to remove pessimism where possible yet to avoid optimism by considering worst case scenarios where it is not feasible to consider exact or probabilistic performance. The pessimism is not extreme; the analysis matches results obtained by accurate simulation very closely.

The standard industrial practice of merely applying the standard schedulability analysis [18] is not sufficient to be able to guarantee performance in critical systems because it relies on the assumption of a minimum inter-arrival time between faults. This analysis allows that assumption to be removed, instead using a probability distribution to model faults more realistically.

The analysis can be completely automated, execution times of the algorithm were no more than a few seconds for the typical message sets used. Further, no changes are required to the system in order to perform this analysis as it models a standard CAN bus. Tool support can be provided in a way that is compatible with standard practice tools which currently use Tindell's schedulability analysis.

In comparison with previous similar work [10] this analysis does not suffer from the two major sources of pessimism described in section 2 (faults occurring after successful transmission cannot affect the response time of the frame, and the pessimistic assumption regarding the application of the error overhead function to burst errors). Overall, our analysis produces probabilities many orders of magnitude lower (less pessimistic) than previously predicted by Navet.

This analysis can also be applied in other domains, for example in processor scheduling to explore the effect of external events (such as user interaction or data packet arrivals) modelled as random sporadic tasks. The analysis here is based on the non-preemptive model of CAN, but it is trivial to adapt to a pre-emptive model.

5.1 Future Work

In the analysis, we consider only the invocation following the critical instant as this is the worst case. However, the other invocations in the hyper-period may have much less interference than the worst case; therefore this is a source of pessimism. We believe that a similar analysis can be produced which covers all invocations, progress has already been made in this area. Another source of pessimism in the analysis that could be avoided is consideration of faults always causing the maximum impact. The analysis can be extended to consider faults which do not cause maximum interference, such as those that happen during bus idle [7] or when an error frame is already being transmitted.

The fault model used in this analysis is simplistic, consisting of only single-bit errors. Other more complex fault models published for CAN [14, 10] have some realistic characteristics. This analysis may be extended to include more complex fault models by considering the possibility of having bursts of errors and multiple sources of interference. However, we do not consider this extension here because multiple fault sources can often be modelled as one source [5] and the effect of short burst errors is not much worse than single-bit errors due to the considerable overhead of a single bit fault.

Where the bursts are very long (*e.g.* as long as the period of a message), the bus is inaccessible for so long that error recovery by retransmission is probably not appropriate, therefore an analysis based on retransmission has no value. In this situation, alternative error recovery protocols such as can be provided by LST-CAN [4], FTTCAN [1] may be more useful.

5.2 Acknowledgements

This work has been funded by EPSRC (UK), Rolls-Royce and by the Spanish MCYT grant DPI2001-2311-C03-02, which is partially funded by the European Union FEDER program. Thanks also to the reviewers of this paper for their helpful comments.

References

[1] L. Almeida and J. Fonseca. FTT-CAN: A network-centric approach for CAN-based distributed systems. In *IFAC 4th Symposium on Intelligent Components and Instruments for Control Applications*, Buenos Aires, Argentina, Sept 2000.

[2] I. J. Bate. *Scheduling and Timing Analysis for Safety Critical Real-Time Systems*. PhD thesis, Dept of Computer Science, University of York, York, YO10 5DD, 1999.

[3] Bosch, Postfach 50, D-700 Stuttgart 1. *CAN Specification*, version 2.0 edition, 1991.

[4] I. Broster and A. Burns. Timely use of the CAN protocol in critical hard real-time systems with faults. In *Proceedings of the 13th Euromicro Conference on Real-time Systems*, Delft, The Netherlands, June 2001. IEEE.

[5] M. J. Buckingham. *Noise in Electronic Devices and Systems*. Series in Electrical and Electronic Engineering. Ellil Horwood/Wiley, 1983.

[6] J. Charzinski. Performance of the error detection mechanisms in CAN. In *Proceedings of the 1st International CAN Conference*, pages 20–29, Mainz, Sept 1994.

[7] H. Hansson, C. Norström, and S. Punnekkat. Integrating reliability and timing analysis of CAN-based systems. In *Workshop on Factory Communications Systems*, Porto, Portugal, Sept 2000. IEEE.

[8] International Standards Organisation. *ISO 11898. Road Vehicles – Interchange of digital information – Controller area network (CAN) for high speed communication*, 1993.

[9] R. T. McLaughlin. EMC susceptibility testing of a CAN car. SAE Technical Paper 932886, SAE, 1993. http://www.warwick.ac.uk/devicenet/publications.htm.

[10] N. Navet, Y.-Q.Song, and F. Simonot. Wost-case deadline failure probability in real-time applications distributed over controller area network. *Journal of Systems Architecture*, 46(1):607–617, 2000.

[11] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat. Using bit-stuffing distributions in CAN analysis. In *IEEE Real-Time Embedded Systems Workshop at the Real-Time Systems Symposium*, London, UK, 2001.

[12] L. M. Pinho, F. Vasques, and E. Tovar. Integrating inaccessibility in response time analysis of CAN networks. In *Proceedings of the 3rd IEEE Workshop On Factory Communication Systems*, pages 77–84, Porto, Portugal, September 2000.

[13] J. Proenza and J. Miro-Julia. MajorCAN: A modification to the Controller Area Network protocol to achieve Atomic Broadcast. In *IEEE Int. Workshop on Group Communications and Computations. IWGCC. Taipei, Taiwan*, April 2000.

[14] S. Punnekkat, H. Hansson, and C. Norström. Response time analysis under errors for CAN. In *Proceedings of RTAS 2000*, pages 258–265, Washington DC, 2000. IEEE.

[15] J. Rufino. Redundant CAN architectures for dependable communication. Technical Report CSTC Technical Report RT-98-02, Instituto Superior Tecnico, NavIST Group, Lisboa, Portugal, 1998.

[16] SAE. Class C application requirement considerations. Technical Report J2056/1, Society of Automotive Engineers, 1993.

[17] M. J. Scholfield. Controller area network (CANbus). http://www.mjschofield.com/, 2002.

[18] K. Tindell, A. Burns, and A. J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[19] J. Unruh, H. Mathony, and K. H. Kaiser. Error detection capabilities of the CAN protocol. Technical report, Robert Bosch GmbH, Stuttgart, Germany, Dec. 1989.