Hard Real-Time in a Soft World

Hans Hansson, Christer Norström and Sasikumar Punnekkat

Mälardalen Real-Time Research Centre,
Department of Computer Engineering
Mälardalen University, Västerås, Sweden
www.mrtc.mdh.se

Abstract

In cost conscious industries, such as automotive, it is imperative for designers to adhere to policies that reduce system resources to the extent feasible, even for safety-critical sub-systems. However, the overall reliability requirement must be both analysable and met. Faults may be either, hardware, software or timing faults. The latter being handled by hard-real time schedulability analysis, which is used to prove that no timing violations will occur. However, from a reliability and cost perspective there is a trade-off between timing guarantees, the level of hardware and software faults, and the per-unit cost. By allowing occasional deadline misses, less costly hardware may be used, while still meeting the overall reliability requirement. Careful analysis is however needed. The main risk/problem is that this type of reasoning is highly dependent on assumptions concerning distributions and independence.

This paper presents a reliability analysis method that considers the effects of faults and timing parameter distributions on schedulability analysis, and its impact on the reliability estimation of the system. In scheduling terms, we will consider a wider set of scenarios/cases than just the worst case considered in hard real-time schedulability analysis. The ideas have general applicability, but the method has been developed with modelling of external interference of automotive CAN buses in mind. We illustrate the method by showing that a CAN-bus interconnected distributed system, subjected to external interference, may be proven to satisfy its timing requirements with a sufficiently high probability, even in cases when the worst-case analysis has deemed it unschedulable.

1 Introduction

The parallel evolution of fault tolerance and real-time realms of research, though have been greatly successful independently, still fail to bring the necessary synergy between the two fields which both are of extreme importance in the design of safety-critical systems. Their mutual dependencies and interactions need to be analysed carefully for achieving predictable performance. In order to bridge the gap between these two areas, several open issues need to be addressed in their totality, a typical one being the effect of faults on schedulability analysis and its impact on the reliability estimation of the system.

The major stumbling block in having an integrated approach is the orthogonal nature of the two factors, viz., the stochastic nature of faults and the deterministic requirements on schedulability analysis. This calls for development of more realistic fault models which capture the nuances of the environment, as well as methods for easy integration of such models into the timing analysis, and finally, a unified and `formal' approach in using these factors in obtaining refined estimates for the reliability of the system.

The main focus of the real-time research community is on hard real-time systems, i.e. in the worst case timing behaviour. 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 [4][5][8][11]. The essence of the analysis is to investigate if the 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 (an exception being [6]).

We have recently [7] developed a model for calculating worst-case latencies of messages under error assumptions, on the Controller Area Network (CAN), which is a popular and predictable communication network extensively used in the automotive industry and elsewhere. This analysis might infer that a given message set is not feasible under worst case fault interferences. This result, though correct, is of limited help to system designers except to prompt them to overdesign the system and waste resources to tackle a situation, which might never happen during the lifetime of the system.

Reliability modelling, on the other hand involves study of fault models, characterisation 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.

When performing schedulability analysis (or any other type of formal analysis) it is important to keep in mind that the analysis is only valid under some specific model assumptions, typically under some assumed ``normal condition", e.g., no hardware failures and a ``friendly" environment. The ``abnormal" situations are typically catered for in the reliability analysis, where probabilities for failing hardware and environmental interferences are combined into a system reliability measure. This separation of deterministic (0/1) schedulability analysis and stochastic reliability analysis is a natural simplification of the total analysis, which unfortunately may introduce quite some pessimism, by assuming that the ``abnormal" is equivalent to failure. Especially for transient errors/failures this may not at all be the case. Consider for instance occasional external interference on a communication link. The interference will lead to transmission errors and subsequent retransmission of messages. The effect will be increased message latencies that may lead to missed deadlines, especially if the interference coincides with the worst case message transmission scenario considered when performing schedulability analysis. In other scenarios, the interference will not increase the worst case message latency, as illustrated Figure 1. The figure shows a system with 3 periodic messages M_1, M_2 and M_3 with descending priorities and with periods (equals to deadlines) of 5, 10 and 20 and worst-case transmission times of 2, 1 and 1 respectively. Assuming an overhead, O=1 for error signalling and recovery (but not including retransmission of the corrupted message), we have shown the effects of 3 different scenarios, corresponding to an external interference hitting the system at points in time. In the first case the error caused by the interference results in a deadline miss for M_2 and M_3. In the second case, though a re-transmission is necessitated, still the message set meets its deadlines, whereas in the third scenario, the error has no effect at all since it falls in a period of inactivity of bus.

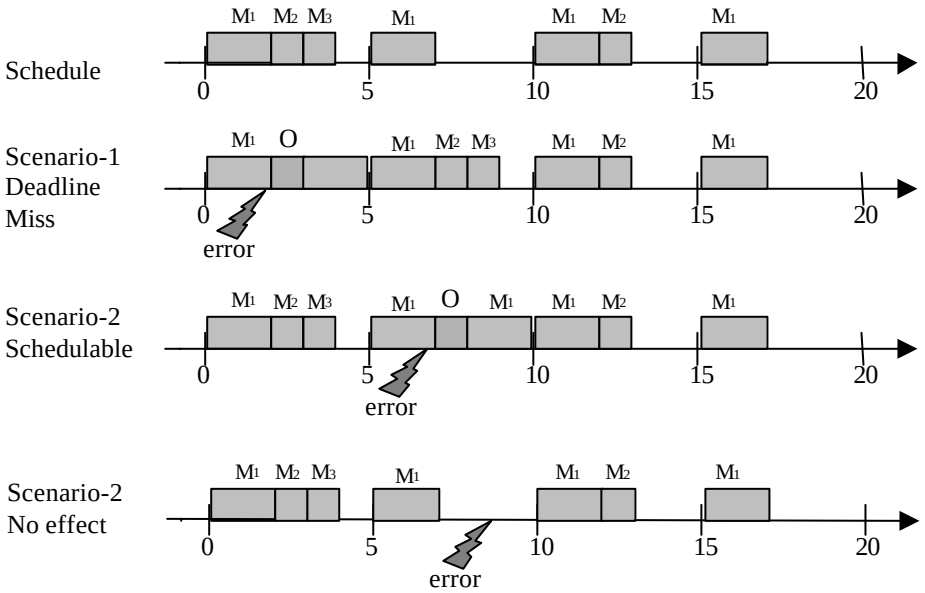

Figure 1: Dependency of Effects of Faults on Phasings

This simple example shows that there are situations (scenarios) when system requirements (e.g. deadlines) are not violated by the ``abnormal". Hence, there is a potential for obtaining a more accurate and tight reliability analysis by considering the likelihood of the ``abnormal" actually causing a deadline violation, i.e., by integrating schedulability and reliability analysis.

Considering the cost-consciousness of industry, the pessimism in the hard real-time analysis, and the low probability of the considered worst-case actually occurring, it is additionally tempting to reduce the resources requires by also trading the absolute timing guarantees under fault-free conditions for reduced reliability (given that the resulting over all reliability is acceptable).

In the method presented in this paper, we will not only considering the worst case values for parameters, such as execution time and periods. Instead we will model their distributions. The analysis is then performed by repeatedly analysing systems with fixed parameter values, obtained by sampling these distributions until required confidence is reached. In doing this we must be careful in validating that assumed independence between distributions are actually satisfied, or alternatively if distributions are dependent, we must faithfully capture the dependencies.

The underlying argument of our work, is that for any system (even the most safety critical one) the behaviour can only be guaranteed up to some level, after which we must resort to reliability analysis and corresponding requirements.

The contributions of this paper are:

- A new approach towards integrating schedulability analysis and reliability models
- A unified framework for holistic analysis of systems' timing and reliability behaviours
- A systematic procedure for obtaining more accurate reliability estimates
- An illustrative example presenting our method

The paper is organised as follows: In Section 2 we introduce our method for considering schedulability in reliability analysis. Section 3 presents our case-study, a distributed automotive control system. Finally, in Section 4 we conclude and outline future directions.

2 The Method

The method we propose assumes an existing task model M with a set of parameters. As an illustration, consider the basic rate-monotonic model introduced by Liu and Layland [12]. The parameters of this model are the periods (T) and execution time requirements (C) of the tasks.

We will partition the set of parameters into two disjoint subsets: the set of deterministic parameters $P_D(M)$ and the set of stochastic parameters $P_S(M)$. Deterministic parameters have fixed values, whereas stochastic parameters have an associated distribution over a specific value domain. Returning to the rate-monotonic example, we can imagine a system for which the task periods are deterministic and the execution times stochastic. We could for instance have a rectangular distribution over the interval $[C_{max}/2, C_{max}]$, where C_{max} is the worst-case execution time used in the original analysis. A concrete example with three tasks is illustrated in.

Original RM model			
Task name	T	C	R
A	5	2	2
B	10	3	5
C	15	4	18*

Extended model with stochastic C			
Task name	T	C	R
A	5	[1,2]	[1,2]
B	10	[1.5,3]	[2.5,5]
C	15	[2,4]	[4.5,18*]

Figure 2: Simple example, showing a RM task set (left) and the corresponding set with stochastic execution time parameter C (right).

We have used exact analysis to calculate the response times R for the task sets in Figure 2. The result, shown in Figure 2, indicate that tasks A and B will always meet their deadlines, whereas task C will miss its deadline (indicated by *) in the worst case considered in the original analysis. (Note that the calculated R-value in case of exceeded deadline is only a lower bound, since the used formula does not correctly capture the case when R exceeds T.) In the stochastic case (to the right in Figure 2), we have calculated ranges of possible R values by simply performed calculations with the extreme parameter values (max and min for all involved parameters, respectively). We can see that task C will sometimes complete within its deadline, sometimes not. The question is now: How often will C miss its deadline?

A naïve approach to answering this question could be to investigate all possible parameter value combinations. If we assume a scheduling granule of 0.1 time units, then it follows that the number of possible combinations of stochastic parameter values are 10*15*20=3000 (since the number of distinguishable execution time intervals for the tasks are 10, 15 and 20, respectively). We could then perform schedulability analysis for all these cases to calculate the fraction of parameter combinations for which C misses its deadline.

This is however not correct. To understand why we need to take a closer look at the schedulability analysis equation (from [4]):

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

In calculating the response time of a task one or more invocations of each higher priority task will interfere (the exact number is given by the ceiling expression in the above formula). In cases with more than one invocation it would be incorrect to use the same execution time value (C_i in the formula) for all invocations, unless our interpretation is that the selected value is an upper bound of the actual execution times. If they, on the other hand, are distributions over actual execution times (which is our intention), then we are only considering a subset of all possible combinations. We could get around this problem by selecting a new value for each invocation. This will increase the number of cases to investigate to $10^6*15^3*20^1$=33 750 000 000 combinations (since we terminate the analysis when the deadline is exceeded it is sufficient to consider invocations up to T_C). This result is however only valid for a specific T_C, since even if all deadlines are shown to be met in the T_C, later deadlines may be violated. As an illustration consider the following example: Assume that the two tasks X and Y have parameters T_X=3, T_Y=4, C_X=[1,2], and C_Y=2. If C_X is equal to 1 in the first invocation, and thereafter 2, we get the schedule in Figure 3 during the first 9 time units:

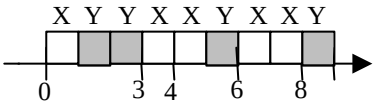

Figure 3: Schedule indicating that Task Y is meets its deadline at time 4, but misses it at time 8.

Hence, Y will miss its deadline at time 8 (with one time unit left to execute), even though analysis from time 0 showed that it is schedulable. We conclude that this type of schedulability analysis with non worst-case execution times only tells us whether the current invocation is schedulable, not anything conclusive about subsequent invocations.

To get around this problem we propose a simulation based approach, rather than using schedulability analysis equations. The idea is to perform extensive simulations of the behaviour. Given strictly periodic tasks, and an initial worst case phasing, it is sufficient to simulate the system during a period equal to the least common multiple (LCM) of the task periods, with independent sampling of successive invocations of the same task. This will as explained above give a state space of 33.750.000.000 combinations for the task set in Figure 2. It should also be noted that this would only give the probability of a missed deadline in an arbitrary LCM. We can from this probability *p* calculate the probability of no missed deadlines during a mission time of length Z by the formula: $1-(1-p)^{\frac{Z}{LCM}}$. This will give a very small probability, unless a very small Z and small *p*.

It should be obvious from the above that we cannot perform complete analysis in more complex cases (more than 33 G combinations to investigate is already too much!). To handle larger state spaces we suggest a partial analysis method based on sampling, i.e. only a subset of the combinations are investigated. This will of course give a result that will be close to the actual probability only with some specific confidence less than 1. For instance, in our simple example with 33.75 G combinations, taking 60.000 samples will with 95% confidence give a probability of no missed deadlines in an LCM in the range [0.9999405,0.9999760].

Our experiments also show that for mission times in the order of hours and LCMs in the order of 100 milliseconds (typical values for many applications), the probability of a single missed deadline during a mission becomes rather high (due to the $1-(1-p)^{\frac{Z}{LCM}}$ formula). For the case-study in Section 3,

which we prove e to be unschedulable with traditional schedulability analysis, we obtain from simulation a probability of 0.02 for at least a single deadline failure in an LCM. If the mission time is 8h (typical value for a vehicle) and the LCM is 120ms (as in our case-study) we get a probability of $1-(1-0.02)^{\frac{28800}{0.12}} \approx 1$ for at least a single deadline failure during a mission. It is however well known (see e.g. [1]) that a control system that will fail due to a single deadline miss is not robust enough to be of much practical use. Rather the system should tolerate single deadline misses, or even multiple deadline misses or more complex requirements on the acceptable pattern of deadline misses. These requirements should of course be derived from the requirements on stability in the control of the external process. By defining a system failure as more than 2 consecutive deadline misses or more than 3 missed deadlines in 15 consecutive periods, we obtain a failure probability in the order of 0.9999 for our case-study in Section 3.

Dependency Issues

In the modelling, the stochastic variables should be very carefully selected to avoid dependencies. There may be several sources of dependencies, including

- apparent dependencies, which are explicitly dependent variables (e.g., those expressed by the schedulability equations), such as
  - response times being dependent on execution times and frequencies of higher priority tasks, and
  - jitter being dependent on response times of preceding tasks in a transaction, and
- subtle dependencies, such as
  - couplings between execution times of tasks due to functional dependencies (this includes dependencies between successive invocations of the same task), and
  - indirect dependencies via the controlled process.

Clearly, all apparent dependencies should be avoided if the dependency cannot be accurately captured. In the case of subtle dependencies, we do not *a priori* know if the variables are dependent or not. If independence is assumed, we recommend monitoring of the variables to validate that they are sufficiently independent, alternatively if independence cannot be concluded, to collect information for a characterisation of the dependencies.

In our case study we will use stochastic variables representing task execution times, and phasings of external interference. All these variables can be considered to be subtle, and we will assume that they are independent. This is however not at all obvious in all cases:

- The execution time of a task can be considered a function of its initial state and input data. In many cases (but not all), the initial state of an invocation is dependent on the task history (i.e. previous invocations and input). Hence, there is probably a dependency between the execution time of successive invocations. There may also be dependencies between execution times of different tasks, e.g. if they share input data or state. Consequently, assuming independence between execution times is rather optimistic.
- The phasings of external interferences relative task executions is clearly independent from other variables, unless the controlled process itself causes the interference, which is very unlikely.

3 The case-study

Our case-study is a simplified automotive control system consisting of two nodes interconnected by a Controller Area Network (CAN) bus (see Figure 4). Two control loops are executing in the system. The first is implemented by a transaction consisting of a sensing task at node A, a message transfer from node A to node B, and a calculating and actuating task at node B, and the second loop is being implemented by a similar transaction in the opposite direction. Furthermore, we will assume background traffic on the bus, which is transmitted at lower priority than the messages of our transactions. There are additionally a single source of external interference, which occasionally induce errors on the bus, leading to retransmissions of corrupted messages. The parameters of our system are summarised in Figure 5 (the details are explained later).

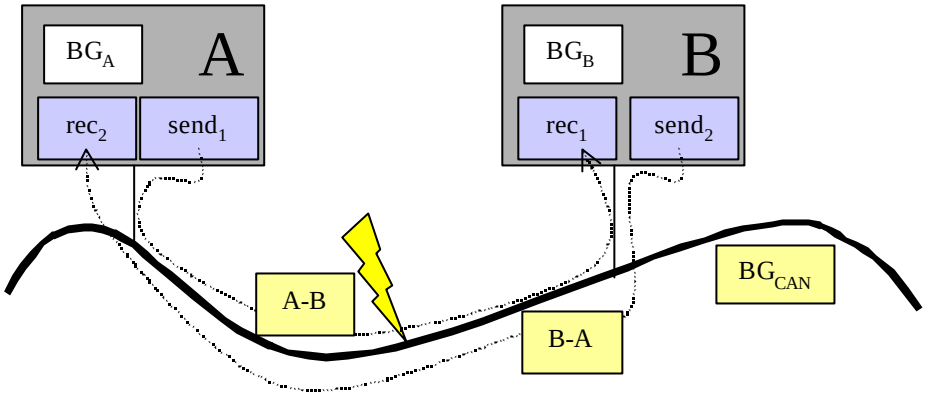

Figure 4: The structure of our simple case-study.

Tasks at node 1			
Name	T	C	Prio
BG_A	2000	[50,300]	H
$SEND_1$	8000	[500,2000]	M
REC_2	(10000)	[1000,2000]	L
Messages sent on the bus			
Name	T	C	Prio
BG_{CAN}	15000	540	L
B-A	(10000)	540	M
A-B	(8000)	540	H

Tasks at node 2				
Name	T	C		Prio
BG_B	2000	[50,200]		H
$SEND_2$	10000	[1000,2000]		M
REC_1	(8000)	[1000,2000]		L
Transactions				
Name	From	Dest	via	Deadline
ATOB	$SEND_1$	REC_1	A-B	6000
BTOA	$SEND_2$	REC_2	B-A	7000
External interference				
	t_f^l	I^l	n^l	T_f^l
	4000	500	4	30 000 000

Figure 5: Summary of parameters for the system in Figure 4.

3.1 The CAN bus

The Controller Area Network (CAN) is a broadcast bus designed to operate at speeds of up to 1 Mbps. Data is transmitted in messages containing between 0 and 8 bytes of data. An 11-bit identifier is associated with each message. The identifier serves two purposes: (1) assigning a priority to the message, and (2) enabling receivers to filter messages.

CAN is a collision-detect broadcast bus, which uses deterministic collision resolution to control access to the bus. 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 message 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 message must be transmitting the highest priority queued message, and hence can start transmitting the body of the message.

3.2 Schedulability Analysis

Tindell et al. [9] [10]have developed schedulability analysis for the CAN bus, which we have extended with a more general fault model [7]. Tindell et al. [2] have also developed holistic analysis which can be used to solve the type of mutually dependent equations which arise in systems with transactions in multiple directions, as in our case.

We will now summarise the relevant theory and present the schedulability analysis for our case-study. It should be noted that this analysis considers the worst-case combination of system parameters, and if it deem the system schedulable, then all deadlines will always be met (given of course that the underlying model assumptions concerning perfect hardware etc. are not violated).

CAN-bus analysis

Tindell et al. [9][10] present analysis to calculate the worst-case latencies of CAN messages. This analysis is based on the standard fixed priority response time analysis for CPU scheduling [7].

Calculating the response times requires a bounded worst case queuing pattern of messages. The standard way of expressing this is to assume a set of traffic streams, each generating messages with a fixed priority. The worst case behaviour of each stream is to periodically queue messages. In analogue with CPU scheduling, we obtain a model with a set S of streams (corresponding to CPU tasks). Each $s \in S$ is a triple $<P_S, T_S, C_S>$, where P_S is the priority (defined by the message identifier), T_S is the period, and C_S the worst case transmission time of messages sent on stream s. The worst-case latency R_i of a CAN message sent on stream S_i is defined by

$$R_i = J_i + q_i + C_i$$

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

$$q_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j + E(q_i + C_i)$$

where the term B_i is the worst-case blocking time of messages sent on S_i, 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, and $E(q_i+C_i)$ is an error term denoting the time required for error signalling and recovery. The reason for the blocking factor is that transmissions are non-preemptive, i.e., after a bus arbitration has started the message with highest priority among competing messages will be transmitted, even if a message with higher priority is queued before the transmission is completed.

Our previous generalisation

In [7] we present a generalisation of the relatively simplistic error model by Tindell and Burns [9]. Our error model specifically considers multiple sources of errors and the signalling pattern of individual sources, consisting of of shorter or longer bursts, during which no signalling will be possible on the bus.

In this paper we will use a slightly simplified version of the error model introduced in [7], with k sources of interference (with each source l contributing an error term $E_i^l(t)$); each source l interferes by inducing an undefined bus value during a characteristic time period I^l, and patterns of interferences for each source l can independently be specified as a sequence of bursts with period T_f^l, where each group consists of n^l interferences of length I^l and with period t_f^l.

Figure 6 illustrates the interference pattern from a single source with $n^l=3$.

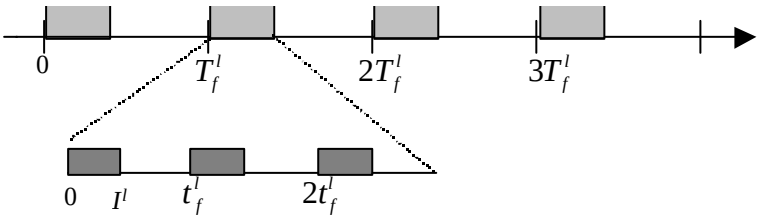

Figure 6: Interference pattern from a single source

We can now define $E_i(t)$ for the case of k sources of interference:

$$E_i(t) = E_i^1(t) \mid E_i^2(t) \mid \ldots \mid E_i^k(t)$$

where

$$E_i^l(t) = Bu^l(t)*(O_i + \max(0, I^l - \tau_{bit}))$$

where the number of interferences until t, $Bu^l(t)$, is given by

$$Bu^l(t) = \left\lfloor \frac{t}{T_f} \right\rfloor * n^l + \min\left(n^l, \left\lceil \frac{t \bmod T_f^l}{t_f^l} \right\rceil \right)$$

Some explanations:

- $\max(0, I^l - \tau_{bit})$ defines the length of I^l exceeding τ_{bit}
- $\left\lfloor \frac{t}{T_f} \right\rfloor$ is the number of full bursts until t.
- $\left\lceil \frac{t \bmod T_f^l}{t_f^l} \right\rceil$ is the number of t_f^l periods that fit in the last (not completed) burst period in t.

We assume that the overheads O_i are given by:

$$O_i = 31*\tau_{bit} + \max_{k \in hp(i) \cup \langle i \rangle} C_k$$

where $31*\tau_{bit}$ is the time required for error signalling in CAN and the max-term denotes the worst-case retransmission time.

Holistic analysis

In analysing our entire distributed system we will use the holistic schedulability analysis developed by Tindell and Clark [2]. The basic idea of this analysis is to solve a set of mutually dependent equations – one for each resource. The equations are, due to transactions spanning several resources, coupled via inheritance of jitter.

We start by presenting the schedulability analysis for the CPUs. We will here assume a very simple model with jitter, but without blocking or deadlines exceeding periods. This gives us the following equation for the response time of a task *i*:

$$R_i = J_i + w_i$$

$$w_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i + J_j}{T_j} \right\rceil C_j$$

To see how the equations are coupled we show the equations for the ATOB transaction:

$$R_{send_1} = J_{send_1} + w_{send_1}$$

$$J_{send_1} = 0$$

$$w_{send_1} = C_{send_1} + \sum_{j \in hp(i)} \left\lceil \frac{w_{send_1} + J_j}{T_j} \right\rceil C_j$$

$$R_{ATOB} = J_{ATOB} + q_{ATOB} + C_{ATOB}$$

$$J_{ATOB} = R_{send_1} - 500$$

$$q_{ATOB} = 540 + \sum_{j \in hp(i)} \left\lceil \frac{q_{ATOB} + J_j + \tau_{bit}}{T_j} \right\rceil C_j + E(q_{ATOB} + C_{ATOB})$$

$$R_{rec_1} = J_{rec_1} + w_{rec_1}$$

$$J_{rec_1} = R_{ATOB} - 340$$

$$w_{rec_1} = C_{rec_1} + \sum_{j \in hp(i)} \left\lceil \frac{w_{rec_1} + J_j}{T_j} \right\rceil C_j$$

Analysis Results

Using the schedulability analysis tool FPSCalc [3] and the parameters in Figure 5, we have performed schedulability analysis of our case-study. The results are reported in Figure 7. We note that both transactions ATOB and BTOA miss their deadlines, since all response times are larger than the corresponding deadlines, 6000 and 7000, respectively.

Schedulability analysis, assuming WCET with no error source interferring	
Name	R
ATOB	8280
BTOA	8920
Schedulability analysis, assuming WCET with one error source interferring.	
Name	R
ATOB	9440
BTOA	10080

Figure 7: Schedulability analysis results

3.3 Simulation

To derive reliability estimates we will perform simulations of the system behaviour. In this simulation, we will assume:

1. Worst-case phasings of message and task queuings at time 0 in the LCM (actually this could be at any time, so why not choose 0). This introduces some pessimism, since the worst case might not occur in every LCM.
2. Random phasings of interference. This can be expressed as an offset from the beginning of the LCM to when the first interference hits. For each source that hits the LCM, such an offset should be "sampled".
3. Sampling of execution times each time a task is released. The execution time is assumed to have a rectangular distribution ranging from best case to worst case.
4. Perfectly synchronised clocks.

The simulator uses a separate ready queue for each node and a single shared ready queue for the CAN-bus. Each message or task can be in one of the three states: waiting, ready, or executing. The nodes are pre-emptively scheduled, while in a fault free CAN-bus message transmissions are non-preemptive. However, in case of errors, a message can be interrupted and has then to be re-queued. In this case-

study we will measure the number of missed deadlines for each transaction for the following different cases:

- Exhaustive simulation assuming WCET with and without error source interference.
- Sampling of execution time (C) with and without error source interference.
- Relaxed failure semantics with and without error source interference, i.e., defining a failure as either 2 or more consecutive deadline misses or more than 3 faults per 15 consecutive transactions.

3.3.1 Simulation Results

In this example we show how to go from a non-schedulable system to a system with a rather high reliability by sampling execution times and relaxing the failure semantics.The simulation shows that assuming maximum execution time when performing schedulability analysis is very pessimistic. If we look at transaction ATOB we increase the number of transaction fulfilling their deadlines from 60% to 98% by sampling of the execution times. Further, if the failure semantics is relaxed, we achieve an even higher reliability. As mentioned earlier, a system that fails when missing one deadline is not robust.

Exhaustive simulation assuming WCET with no error source interferring					
Name	BCRESP[1]	WCRESP[2]	#trans[3]	#MisD	SatD[4]
ATOB	5340	7540	15	6	0,6
BTOA	5240	7540	12	3	0,75
Exhaustive simulation assuming WCET with one error source interferring.					
Name	BCRESP	WCRESP	#trans	#MisD	SatD
ATOB	5340	8700	268620	181395	0,59691
BTOA	5240	8900	269706	90306	0,74915

Sampling of C with no error source interference.					
Name	BCRESP	WCRESP	#trans	#MisD	SatD
ATOB	2250	7014	15000	299	0,980067
BTOA	2738	7150	12000	2	0,999833
Sampling of C with error source interference.					
Name	BCRESP	WCRESP	#trans	#MisD	SatD
ATOB	2250	7194	15000	310	0,979333
BTOA	2738	7150	12000	2	0,999833

Relaxing the failure semantics to define a failure as either 2 or more consecutive deadline misses or more than 3 faults per 15 consecutive transactions, we get the following figures:

Sampling of C with no error source interference.					
Name	BCRESP	WCRESP	#trans	#MisD	NoFail
ATOB	2250	7194	15000	3	0,999867
BTOA	2738	7150	12000	0	1
Sampling of C with error source interference.					
Name	BCRESP	WCRESP	#trans	#MisD	NoFail
ATOB	2250	7194	15000	2	0,999800
BTOA	2738	7150	12000	0	1

To get our final reliability estimate we additionally have to consider the interarrival time of successive inference bursts. This is given by the T_f^1 period. Within each such period there will be n^1 bursts with

[1] Best case response time monitored.
[2] Worst case response time monitored.
[3] The total number of executions of a transaction.
[4] Number of fulfilled deadlines.

period t_f^l, during these periods only transmissions will be subjected to interference. Hence, the reliability is defined by the following formula:

$$\frac{T_f^l - t_f^l * n_l}{T_f^l} * NoFail_{\text{no\_interference}} + \frac{t_f^l * n_l}{T_f^l} * NoFail_{\text{interference}}$$

which in our case gives an over all reliability estimate of 0.999867.

Note that, if we hypothetically would have had a schedulable interference-free system, we get 0.999999893, corresponding to a failure probability of 10^{-7}.

3.3.2 Confidence

The confidence calculation is based on the central limit theorem [13], which gives us the possibility to calculate the average of *n* random variables. When *n* becomes large, the distribution tends to the standard normal. In our case we have one random variable for each sample. By taking enough samples we can use the theorem and calculate a confidence interval.

Assuming a confidence of 95% and 1000 samples we get an average of 0,9793333 and a confidence interval of [0,9771856, 0,981481] for the case**:** "Sampling of WCET with error source interference".

4 Discussions/ Conclusions

We have presented a reliability analysis method, which combines reliability and schedulability analysis. The key components of this method are the introduction of distributions of timing parameters and simulation to obtain reliability estimates. In applying this method to a simple case-study we have shown that a system which was deemed unschedulable by traditional analysis can be proven to meet its timing requirements with a relatively high probability.

We have presented early and preliminary results from an ongoing effort aiming at providing designers with well founded arguments for making reasonable choices in the trade-off between cost, real-time guarantee, and reliability.

5 References

[1] Törngren M. Fundamentals of Implementing Real-Time Control Applications in Distributed Computer Systems, Real-Time Systems, 14, 219-250 (1998), Kluwer Academic Publisher.

[2] Tindell K., J. Clark, Holistic Schedulability Analysis for Distributed Hard Real-time Systems. Technical Report YCS197, Real-Time Systems Research Group, Univ. of York, 1993.

[3] FpsCalc. Available athttp://www.docs.uu.se/~ebbe/realtime/marble_sorter/remote.html

[4] 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.

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

[6] A. Burns, S. Punnekkat, L. Strigini, and D.R. Wright. Probabilistic scheduling guarantees for fault-tolerant real-time systems. Proceedings of DCCS-7,IFIP International Working Conference onDependable Computing for Critical Applications, California, January 1999.

[7] S. Punnekkat, H. Hansson, and C. Norstr. om. Response Time Analysis under Errors for CAN. Proceedings of IEEE Real-Time Technology and Applications Symposium(RTAS), page To appear, June 2000.

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

[9] 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 ofYork, June 1994.

[10] 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.

[11] J. Xu and D. L. Parnas. Priority scheduling versus pre-run-time scheduling. Real-Time Systems Journal, 18(1), January 2000.

[12] C. L. Liu and J. W. Layland, *Scheduling Algorithms for Multiprogramming in aard Real-Time Environment*, JACM 20(1):46-61, 1973.

[13] P. Newbold, *Statistics for Business and Economics. Prentice-Hall International*, 1988.