Probabilistic Scheduling Guarantees for Fault-Tolerant Real-Time Systems

A. Burns and S. Punnekkat
Real-Time Systems Research Group
Department of Computer Science
University of York, UK

L. Strigini and D. R. Wright
Centre for Software Reliability
City University, UK

Abstract

Hard real-time systems are usually required to provide an absolute guarantee that all tasks will always complete by their deadlines. In this paper we address fault tolerant hard real-time systems, and introduce the notion of a probabilistic guarantee. Schedulability analysis is used together with sensitivity analysis to establish the maximum fault frequency that a system can tolerate. The fault model is then used to derive a probability (likelihood) that, during the lifetime of the system, faults will not arrive faster than this maximum rate. The framework presented is a general one that can accommodate transient 'software' faults, tolerated by recovery blocks or exception handling; or transient 'hardware' faults dealt with by state restoration and re-execution.

1 Introduction

Scheduling work in hard real-time systems is traditionally dominated by the notion of absolute guarantee. Static analysis is used to determine that all deadlines are met even under the worst-case load conditions. With fault-tolerant hard real-time systems this deterministic view is usually preserved even though faults are, by their very nature, stochastic. No fault tolerant system can, however, cope with an arbitrary number of errors in a bounded time. The scheduling guarantee is thus predicated on a fault model. If the faults are no worse than that defined in the fault model then all deadlines are guaranteed. The disadvantage of this separation of scheduling guarantee and fault model is that it leads to simplistic analysis; either the system is schedulable or it is not.

In this paper we bring together scheduling issues and errors to justify the notion of a *probabilistic guarantee* even for a hard real-time system. By 'probabilistic guarantee' we mean a scheduling guarantee with an associated probability. Hence, a guarantee of 99.95% does not mean that 99.95% of deadlines are met. Rather it implies that the probability of *all* deadlines being met during a given period of operation is 99.95%. Instead of starting with the fault model and using scheduling tests to see if this is feasible, we start with the scheduling analysis to derive a threshold interval between errors that can be tolerated and then employ the fault model to assign a probability to this threshold value.

To provide the flexibility needed to program fault tolerance, fixed priority preemptive scheduling will be used [13]. The faults of interest are those that are transient. Castillo *at al* [6] in their study of several systems indicate that the occurrences of transient faults are 10 to 50 times more frequent than permanent faults. In some applications this frequency can be quite large; one experiment on a satellite system observed 35 transient faults in a 15 minute interval due to cosmic ray ions [5].

We attempt to keep the framework as general as possible by accommodating 'software' faults tolerated by either exception handling or some form of recovery block, and 'hardware' faults dealt with by state restoration and re-execution. Error latencies will be assumed to be short.

Other authors have studied the probability of meeting deadlines in fault-tolerant systems. However, only some facets of this problem have been considered. For instance, Hou and Shin [9] have studied a related problem, the probability of meeting deadlines when tasks are replicated in a hardware-redundant system. However, they only consider permanent faults without repair or recovery. A similar problem was studied by Shin et al [18]. Kim et al [12] consider another related problem: the probability of a real-time controller meeting a deadline when subject to permanent faults with repair.

The rest of the paper is organised as follows. Section 2 briefly describes the scheduling analysis that is applicable to non-fault-tolerant systems. Section 3 presents the fault model and the framework for the subsequent analysis. In Section 4 the scheduling analysis for a fault tolerant system is presented. This enables the threshold fault interval (TFI) to be derived. Section 5 then uses the fault model and the TFI to assign a probability to the threshold. Conclusions are presented in Section 6.

2 Standard Scheduling Analysis

For the standard fixed priority approach, it is assumed that there is a finite number (N) of tasks (τ_1 .. τ_N). Each task has the attributes of minimum inter arrival time, T, worst-case execution time, C, deadline, D and priority P. Each task undertakes a potentially unbounded number of invocations; each must be finished by the deadline (which is measured relative to the task's invocation/release time). All tasks are deemed to start their execution at time 0. We assume a single processor platform and restrict the model to tasks with $D \leq T$. For this restriction, an optimal set of priorities can be derived such that $D_i < D_j \Rightarrow P_i > P_j$ for all tasks i, j [15]. Tasks may be periodic or sporadic (as long as two consecutive releases are separated by at least T). Once released, a task is not suspended other than by the possible action of a concurrency control protocol surrounding the use of shared data. A task, however, may be preempted at any time by a higher priority task. System overheads such as context switches and kernel manipulations of delay queues etc can easily be incorporated into the model [11, 4] but are ignored here.

The worst-case response time (completion time) R_i for each task (i) is obtained from the following [10, 1]:

$$R_i \;=\; C_i \;+\; B_i \;+\; \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{1}$$

where $hp(i)$ is the set of higher priority tasks (than i), and B_i is the maximum blocking time caused by a concurrency control protocol protecting shared data.

The most common and effective concurrency control protocol assigns a ceiling priority to each shared data area. This ceiling is the maximum priority of all tasks that use the shared data area. When a task enters the protected object that contains the shared data, its priority is temporarily increased to this ceiling value. As a consequence (on a single processor system):

1. Mutual exclusion is assured (by the protocol itself).
2. Each task is only blocked once during each invocation.
3. Deadlocks are prevented (by the protocol itself).

The value of B_i is simply the maximum computation time of any protected object that has a ceiling equal or greater than P_i and is used by a task with a priority lower than P_i.

To solve equation (1) a recurrence relation is produced:

$$r_i^{n+1} \;=\; C_i \;+\; B_i \;+\; \sum_{j \in hp(i)} \left\lceil \frac{r_i^n}{T_j} \right\rceil C_j \tag{2}$$

where r_i^0 is given an initial value of C_i (although more efficient initial values can be found). The value r^n can be considered to be a computational window into which an amount of computation C_i is attempting to be placed. It is a monotonically non-decreasing function of n. Note that when r_i^{n+1} becomes equal to r_i^n then this value is the worst-case response time, R_i [4]. However if r_i^n becomes greater than D_i then the task cannot be guaranteed to meet its deadline, and the full task set is thus unschedulable.

Table 1 describes a simple 4 task system, together with the response times that are calculated by equation (2). Priorities are ordered from 1, with 4 the lowest value, and blocking times have been set to zero for simplicity. Scheduling analysis is independent of time units and hence simple integer values are used (they can be interpreted as milliseconds).

To illustrate how these values are obtained consider τ_4; r_i^0 is given the initial value of 30, r_i^1 is then just the addition of all the computation times (30 + 35 + 25 + 30 = 120), so r_i^2 is assigned 120. With this value τ_1 gives rise to another hit (of 30)

Task	P	T	C	D	B	R	Schedulable
τ_1	1	100	30	100	0	30	TRUE
τ_2	2	175	35	175	0	65	TRUE
τ_3	3	200	25	200	0	90	TRUE
τ_4	4	300	30	300	0	150	TRUE

Table 1: Example Task Set

and hence r_i^3 is 150. This value is then stable and hence is the required response time.

All tasks are released at time 0. For the purpose of schedulability analysis, we can assume that their behaviour is repeated every LCM, where LCM is the least common multiple of the task periods. When faults are introduced it will be necessary to know for how long the system will be executing. Let L be the lifetime of the system. For convenience we assume L is an integer multiple of the LCM. This value may however be very large (for example LCM could be 200ms, and L fifteen years!).

3 Fault Model

We assume that a single transient fault will cause just one error, and that this error will manifest itself in just a single task. With 'software' faults this is a reasonable assumption. With 'hardware' faults we are concerned with errors that manifest themselves in the processing unit (including internal busses, cache etc) rather than in memory where the error latencies may be very large. We assume that only the executing task is affected[1]. Faults that affect the kernel must either be masked or lead to permanent damage that can only be catered for by replication at the system level. To make the subsequent analysis simpler we assume perfect error recognition coverage; a probabilistic (non zero) measure of coverage could be used with a straightforward effect upon the analysis.

We make the common *homogeneous Poisson process* (HPP) assumptions that the fault arrival rate is constant and that the distribution of the fault-count for any fixed time interval can be approximated using a Poisson probability distribution. This is an appropriate model for a random process where the probability of an event does not change with time and the occurrence of one fault event does not affect the probability of another such event. A HPP process depends only on one parameter, viz, the expected number of events, λ, in unit time; here events are transient faults with $\lambda = 1/MTBF$, where $MTBF$ is the Mean Time Between transient Faults[2].

[1] An alternative model, namely all non-terminated tasks are affected could also have been used. This would make no fundamental difference to the analysis but would complicate the scheduling equations used in Section 4.

[2] MTBF usually stands for mean time between failures, but as the systems of interest are fault tolerant many faults will not cause system failure. Hence we use the term MTBF to model the arrival

Per the definition of a Poisson Distribution,

$$Pr_n(t) = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

gives the probability of n events during an interval of duration t. If we take an event to be an occurrence of a transient fault and Y to be the random variable representing the number of faults in the lifetime of the system (L), then the probability of zero faults is given by

$$Pr(Y = 0) = e^{-\lambda L}$$

and the probability of at least one fault

$$Pr(Y > 0) = 1 - e^{-\lambda L}$$

Other useful values are:

$$\begin{aligned} Pr(Y = 1) &= e^{-\lambda L}\lambda L \\ Pr(Y < 2) &= e^{-\lambda L}(1 + \lambda L) \end{aligned} \tag{3}$$

We are concerned, in this paper, with the probability of the system being schedulable. We shall write $Pr(S)$ and $Pr(U)$ to denote the probability of schedulability and unschedulability. Of course $Pr(S) = 1 - Pr(U)$.

The analysis given in the next section will determine the threshold fault interval. This gives the sustainable frequency at which faults can occur and the system still meet all its deadlines. Let this frequency be represented by the minimum time interval allowed between faults, T_F. It follows that if W is the shortest interval between fault arrivals during a mission then[3]

$$\begin{aligned} Pr(U) = {} & Pr(U|\textit{no faults}).Pr(\textit{no faults}) \\ & + Pr(U|W{\geq}T_F \textit{ and there are faults}).Pr(W{\geq}T_F \textit{ and there are faults}) \\ & + Pr(U|W{<}T_F \textit{ and there are faults}).Pr(W{<}T_F \textit{ and there are faults}) \end{aligned}$$

Since we are dealing with systems which are schedulable 'under no faults' we can assume $Pr(U|\textit{no faults})$ is zero. Also T_F has been defined so that $Pr(U|W \geq T_F)$ is zero. Hence

$$Pr(U) = Pr(U|W < T_F).Pr(W < T_F)$$

In this paper we will make the conservative assumption that $Pr(U|W < T_F)$ is one. And hence we are left with the evaluation of $Pr(W < T_F)$, i.e. the probability

of transient faults.

[3]To simplify notation and avoid the need to mention special cases in the remainder of the paper, we will regard the events of *no faults* and *one fault* as being subsets of the event $W{\geq}T_F$. The simplest formal mechanism to achieve this is to define W to be an improper random variable, taking the value $W{=}\infty$ when there is exactly one fault, or no faults at all, during the system lifetime. Thus W is always realised with some value, so that $Pr(W{<}T_F) + Pr(W{\geq}T_F) = 1$.

that at least two faults arrive so close together in time that they cannot both be tolerated. This is done in Section 5. Although this assumption is conservative (and hence safe) it is clearly possible to give less pessimistic values. The above formulation will allow such values to be combined with the estimates of $Pr(W < T_F)$ given in Section 5.

Issues concerned with implementing the features suggested by the Fault Model are well addressed by Fetzer and Cristian[8].

Typical Values of Key Parameters

Before proceeding with the analysis it is worth noting the ranges in value of the key parameters of the model. In most applications of interest, the "lifetime" over which a probability of failure is required is the duration of one mission. Mission times for civil aircraft are typically 3-20 hours, but for satellites 15 years of execution may be expected. The iteration periods for control loops are as short as 20ms, other loops and signals may have T values of a few seconds. Precise values for $MTBF$ are not generally known, but in a friendly operating environment perhaps 100 hours is not unreasonable. In more hostile conditions, 20 seconds may be more typical. Although T_F is derived from the characteristics of the task set under consideration, it is worth noting that very small values are unlikely (as a task will not make progress if it suffers repetitive faults), and faults spaced out beyond the LCM of the task periods will easily be catered for; hence: 200ms $< T_F <$ 5 Seconds. Table 2 summarizes these viable ranges for the key parameters (in hours and hours^{-1}).

Parameter	Range
L	$3 - 10^5$
T	$10^{-6} - 10^{-2}$
λ	$10^{-2} - 10^2$
T_F	$10^{-5} - 10^{-2}$

Table 2: Typical Values of Key Parameters

4 Schedulability Analysis for Fault Tolerant Execution

Let F_k be the extra computation time needed by τ_k if an error is detected during its execution. This could represent the re-execution of the task, the execution of an exception handler or recovery block, or the partial re-execution of a task with checkpoints. In the scheduling analysis the execution of task τ_i will be affected by a fault in τ_i or any higher priority task. We assume that any extra computation for a task will be executed at the task's (fixed) priority.

Hence if there is just a single fault, equation (1) will become [16, 2][4]:

$$R_i \;=\; C_i \,+\, B_i \,+\, \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \,+\, \max_{k \in hep(i)} F_k \tag{4}$$

where $hep(i)$ is the set of tasks with priority equal or higher than τ_i, that is $hep(i) = hp(i) + \tau_i$.

This equation can again be solved for R_i by forming a recurrence relation. If all R_i values are still less than the corresponding D_i values then a deterministic guarantee is furnished.

Given that a fault tolerant system has been built it can be assumed (although this would need to be verified) that it will be able to tolerate a single isolated fault. And hence the more realistic problem is that of multiple faults; at some point all systems will become unschedulable when faced with an arbitrary number of fault events.

To consider maximum arrival rates, first assume that T_f is a known minimum arrival interval for fault events. Also assume the error latency is zero (this restriction will be removed shortly). Equation (4) becomes [16, 2]:

$$R_i \;=\; C_i \,+\, B_i \,+\, \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \,+\, \left\lceil \frac{R_i}{T_f} \right\rceil \max_{k \in hep(i)} F_k \tag{5}$$

Thus in interval (0 R_i] there can be at most $\left\lceil \frac{R_i}{T_f} \right\rceil$ fault events, each of which can induce F_k amount of each computation. The validity of this equation comes from noting that fault events behave identically to sporadic tasks, and they are represented in the scheduling analysis in this way [1]. Note the equation is not exact (but it is sufficient): faults need not always induce a maximum re-execution load.

There is a useful analogy between *release jitter* and error latency. If a fault can lie dormant for time A_f, then this may cause two errors to appear to come closer together than T_f. This will increase the impact of the fault recovery. Equation (5) can be modified to include error latency in the same way that release jitter is incorporated into the standard analysis [1]:

$$R_i \;=\; C_i \,+\, B_i \,+\, \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \,+\, \left\lceil \frac{R_i + A_f}{T_f} \right\rceil \max_{k \in hep(i)} F_k \tag{6}$$

As before, this equation can be solved for R_i by forming a recurrency relationship.

Table 3 gives an example of applying equation (6). Here full re-execution is required following a fault. Two different fault arrival intervals are considered. For one the system remains schedulable, but for the shorter interval the final task cannot be guaranteed. In this simple example, blocking and error latency are assumed to be zero. Note that for the first three tasks, the new response times are less than the shorter T_f value, and hence will remain constant for all T_f values greater than 200.

The above analysis has assumed that the task deadlines, Ds, remain in effect even during a fault handling situation. Some systems allow a relaxed deadline

[4] We assume that in the absence of faults, the task set is schedulable.

Task	P	T	C	D	F	R $T_f = 300$	R $T_f = 200$
τ_1	1	100	30	100	30	60	60
τ_2	2	175	35	175	35	100	100
τ_3	3	200	25	200	25	155	155
τ_4	4	300	30	300	30	275	UNSCH

Table 3: Example Task Set - T_f = 300/200

when faults occur (as long as faults are rare). This is easily accommodated into the analysis.

Limits to Schedulability

Having formed the relation between schedulability and T_f, it is possible to apply sensitivity analysis to equation (6) to find the minimum value of T_f that leads to the system being just schedulable. As indicated earlier, let this value be denoted as T_F (it is the threshold fault interval).

Sensitivity analysis [19, 14, 13, 17] is used with fixed priority systems to investigate the relationship between values of key task parameters and schedulability. For an unschedulable system it can easily generate (using simple branch and bound techniques) factors such as the percentage by which all Cs must be reduced for the system to become schedulable.

Similarly for schedulable systems, sensitivity analysis can be used to investigate the amount by which the load can be increased without jeopardising the deadline guarantees. Here we apply sensitivity analysis to T_f to obtain T_F.

When the above task set is subject to sensitivity analysis it yields a value of T_F of 275. The behaviour of the system with this threshold fault interval is shown in Table 4. A value of 274 would cause τ_4 to miss its deadline.

Task	P	T	C	D	R $T_F = 275$
τ_1	1	100	30	100	60
τ_2	2	175	35	175	100
τ_3	3	200	25	200	155
τ_4	4	300	30	300	275

Table 4: Example Task Set - T_F set at 275

5 Evaluating $Pr(W<T_F)$

We need to calculate the probability that during the lifetime, L, of the system no two faults will be closer than T_F. Two approaches are considered. The attraction of the first is that it shows that a relatively intuitive and uncomplicated approach yields upper and lower bounds on $Pr(W<T_F)$ which, for a wide range of parameter values, provide a maximum approximation error which cannot be much greater than a factor of 3 (since $\frac{\text{upper bound}}{\text{lower bound}} \approx 3$). With the second approach a more cumbersome but exact formulation is derived. Despite the inclusion of this latter exact formulation, we believe that, given that it is often rather the *order of magnitude* of the failure probability that is the primary concern (rather than an exact value), the mathematically significantly easier reasoning of the first, bounding approach retains some importance.

5.1 Upper and Lower Bounds for Evaluating $Pr(W<T_F)$

We are concerned with two faults being closer than T_F over the mission time L. Since in practice $L \gg T_F$ we can assume, without loss of generality, that L is an even integer multiple of T_F. Let *mishap* be the undesirable event of two faults indeed occurring closer than T_F i.e.

$$Pr(\textit{mishap during } L) \equiv Pr(W < T_F)$$

We derive the required upper and lower bounds via the following theorems:

<u>Theorem 1</u> If $L/(2T_F)$ is a positive integer then

$$Pr(\textit{mishap during } L) < 1 + \left[e^{-\lambda T_F}\left(1+\lambda T_F\right)\right]^{\frac{L}{T_F}-1} - 2\left[e^{-2\lambda T_F}\left(1+2\lambda T_F\right)\right]^{\frac{L}{2T_F}}$$

<u>Theorem 2</u> If $L/(2T_F)$ is a positive integer then

$$Pr(\textit{mishap during } L) > 1 - \left[e^{-\lambda T_F}\left(1+\lambda T_F\right)\right]^{\frac{L}{T_F}}$$

<u>Proof of Theorem 1</u>

Let the mission time be split into a series of 'even' time intervals with boundaries $0, 2T_F, 4T_F, \ldots, L$, as shown in Figure 1. Similarly a set of 'odd' intervals starting at times T_F, $3T_F$, $5T_F$, $\ldots$, $L-T_F$ can be defined (extending the lifetime slightly to $L+T_F$, the end point of the last odd interval, by continuing the same HPP fault model). Each set has $L/2T_F$ intervals. Let a mishap be said to lie in an interval if both of its faults occur during that interval. It follows from the geometry of these intervals that

$$\begin{aligned} \textit{mishap during } L \;\Rightarrow\; & \textit{mishap in some even interval[s]}, \\ & \text{or } \textit{mishap in some odd interval[s]} \end{aligned}$$

This property comes directly from the definition of the intervals; if a mishap (two faults closer than T_F) occurs it must lie in either an even or an odd interval[5].

$$\begin{aligned} Pr(\textit{mishap during } L) \;&<\; Pr(\textit{mishap in some even interval[s]} \\ &\quad \text{or } \textit{mishap in some odd interval[s]}) \end{aligned}$$

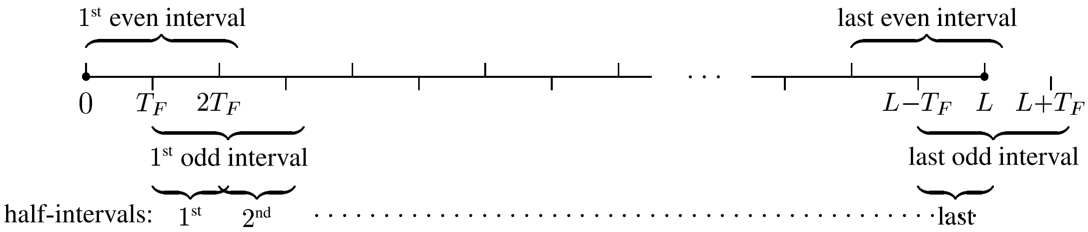

Figure 1: Definitions of *'even intervals', 'odd intervals'* and *'half-intervals'*

Actually the intersection of the two events on the right hand side has non-zero probability. One way that they can occur together is that a single mishap could lie in the overlap between an even and an odd interval. Call these overlaps 'half-intervals': they are of length T_F, and there are[6] $\frac{L}{T_F}-1$ of them, respectively starting at times T_F, $2T_F, \ldots$, $L-T_F$. So from the basic axioms of probability

$$\begin{aligned} Pr(\textit{mishap in some even interval[s]}, \quad \textit{or} \quad & \textit{mishap in some odd interval[s]}) < \\ & Pr(\textit{mishap in some even interval[s]}) \; + \\ & Pr(\textit{mishap in some odd interval[s]}) \; - \\ & Pr(\textit{mishap in some half-interval[s]}) \end{aligned}$$

Now, given the symmetry of the construction and the HPP process assumption,

$$Pr(\textit{mishap in some even interval[s]}) = Pr(\textit{mishap in some odd interval[s]})$$

Hence

$$\begin{aligned} Pr(\textit{mishap during } L) < 2Pr(&\textit{mishap in some even interval[s]}) \\ - \; Pr(&\textit{mishap in some half-interval[s]}) \end{aligned} \tag{7}$$

The event "mishap in a particular even interval" is independent of events in all other even intervals, and it has the same probability for every even interval. Thus

$$Pr(\textit{mishap in some even interval[s]}) = 1 - Pr(\textit{no mishap in } 2T_F)^{\frac{L}{2T_F}}. \tag{8}$$

[5]The difference between the two sides here is the (normally tiny since $L \gg T_F$) probability that the *second* fault of the *first* mishap occurs during $L < t \leq L + T_F$.

[6]Our term 'half-interval' *excludes* the early half of the first even interval, and the latter half of the last odd interval, since they do not arise as overlaps.

For an interval of length $2T_F$ not to contain a mishap, it is sufficient (but not necessary) that it contain 0 or 1 fault. Hence, from equation (3)

$$Pr(\textit{no mishap in } 2T_F) > e^{-2\lambda T_F}\left(1 + 2\lambda T_F\right). \tag{9}$$

Combining equations (8) and (9) yields

$$Pr(\textit{mishap in some even interval[s]}) < 1 - \left[e^{-2\lambda T_F}\left(1 + 2\lambda T_F\right)\right]^{\frac{L}{2T_F}}. \tag{10}$$

By a similar argument for the half-intervals

$$Pr(\textit{mishap in some half-interval[s]}) = 1 - \left[e^{-\lambda T_F}\left(1 + \lambda T_F\right)\right]^{\frac{L}{T_F}-1}, \tag{11}$$

and now combining equations (7), (10), and (11) delivers the theorem statement. □

Proof of Theorem 2

In a similar way to the previous proof, consider the series of intervals of length T_F starting at times 0, T_F, $2T_F$, $3T_F$, $\ldots$, $L{-}T_F$. There are $\frac{L}{T_F}$ of these, a mishap in any one of which implies a mishap during L (but not vice versa). Hence

$$Pr(\textit{mishap during } L) > Pr(\textit{mishap in some interval[s]})$$

but

$$Pr(\textit{mishap in some interval[s]}) = 1 - Pr(\textit{no mishap in any interval})$$

Proof follows directly (as in proof of Theorem 1).
□

Both the upper and lower (exact) bounds are in mathematically non-intuitive forms, but simple approximations can be derived for most of the parameter range within which the probability of mishap is small enough to be of interest.

Corollary 3

An approximation for the upper bound on $Pr(W{<}T_F)$ given by Theorem 1 is $\frac{3}{2}\lambda^2 LT_F$, provided that λT_F, $\lambda^2 LT_F$ are small, and $L{\gg}T_F$.

Corollary 4

An approximation for the lower bound on $Pr(W{<}T_F)$ given by Theorem 2 is $\frac{1}{2}\lambda^2 LT_F$, provided only that λT_F, $\lambda^2 LT_F$ are small.

Proof of Corollary 3

The term $e^{-\lambda T_F}\left(1 + \lambda T_F\right)$ can be approximated by a Taylor series, where terms $(\lambda T_F)^3$ and beyond are ignored. Thus

$$e^{-\lambda T_F}\left(1 + \lambda T_F\right) \approx 1 - \frac{\lambda^2 T_F^2}{2}$$

Another approximation comes from noting that for small xz^2

$$\left(1 - \frac{z^2}{2}\right)^x \approx 1 - \frac{xz^2}{2}$$

where terms z^4 and higher powers of z can be ignored. Hence, under assumptions λT_F, $\lambda^2 L T_F$ small, and $L \gg T_F$, we can write

$$\left[e^{-\lambda T_F}\left(1+\lambda T_F\right)\right]^{\frac{L}{T_F}-1} \approx \left(1 - \frac{\lambda^2 T_F^2}{2}\right)^{\frac{L}{T_F}-1} \approx 1 - \frac{L\lambda^2 {T_F}^2}{2T_F} \tag{12}$$

Under just the first assumptions that λT_F, $\lambda^2 L T_F$ are small we have equally

$$\left[e^{-2\lambda T_F}\left(1+2\lambda T_F\right)\right]^{\frac{L}{2T_F}} \approx 1 - \frac{L\lambda^2 {T_F}^2}{T_F} \tag{13}$$

Applying (12) and (13) to the conclusion of Theorem 1, Corollary 3 is proved.
□

Corollary 4 follows by a similar argument.

Strictly, the bounds in Theorems 1 and 2 have only been proved here for L an even multiple of T_F. However, the realistic assumption $L \gg T_F$ allows the approximations given in the two corollaries still to extend to other values of T_F and L. In fact, where $\frac{L}{2T_F}$ *is* an exact integer, this $L \gg T_F$ assumption is not actually required, either for the derivation of the exact bounds in Theorems 1 and 2, nor for the *lower*-bound approximation of Corollary 4. For high accuracy in Corollary 4, we need only the assumption of small λT_F, $\lambda^2 L T_F$. Corollary 3 is the exception, relying on the $L \gg T_F$ assumption at one place in its derivation: the exponent to the square-bracketed term in (12) is 'out by 1' and we needed $\frac{T_F}{L}$ to be small in order to justify effectively ignoring this fact. For very short mission times, such that we do not have $L \gg T_F$, we can in fact 'retreat slightly' to a slacker upper bound for $Pr(\textit{mishap during } L)$ by using 1 as an upper bound for this square-bracketed term in a modified version of Theorem 1, thus avoiding the awkward exponent $\frac{L}{T_F} - 1$. Then, for positive integral $\frac{L}{2T_F}$, the resulting equivalent of Corollary 3 produces an accurate approximation $2\lambda^2 L T_F$ to this slacker upper bound on $Pr(W < T_F)$, *without* any requirement that $L \gg T_F$; i.e., under the assumptions only that λT_F, $\lambda^2 L T_F$ are small, and that $\frac{L}{2T_F}$ is a positive integer, but without now the requirement that $L \gg T_F$, the methods of this subsection are able to provide bounds on $Pr(W < T_F)$ which are approximately in the ratio $\frac{\text{upper bound}}{\text{lower bound}} \approx 4$.

The important upper bound approximation of Corollary 3 can be written in the form $\frac{3}{2}(\lambda L)(\lambda T_F)$. It will often be the case that $\lambda T_F < 10^{-2}$; indeed this constraint allowed the approximations to deliver useful values. But λL can vary quite considerably from 10^{-2} or less in friendly environments to 10^3 or more in long-life, hostile domains. Clearly, low probability levels for this latter case will be extremely difficult to achieve by the scheduling approach defined in this paper.

	λ		
L	1	10^{-2}	10^{-4}
1	$1.1{\times}10^{-4}$	$1.1{\times}10^{-8}$	$1.1{\times}10^{-12}$
10^1	$1.1{\times}10^{-3}$	$1.1{\times}10^{-7}$	$1.1{\times}10^{-11}$
10^2	$1.1{\times}10^{-2}$	$1.1{\times}10^{-6}$	$1.1{\times}10^{-10}$
10^4	1	$1.1{\times}10^{-4}$	$1.1{\times}10^{-8}$

Table 5: Upper bound on Non-Schedulability due to Faults.

The example introduced in Section 4 had a T_F value of 275ms. Table 5 gives the upper bound on the probability guarantee for various values of λ and L.

When $\lambda L{<}10^{-2}$, λL approximates the probability of any fault happening during the mission of duration L. So, $\frac{2}{3}(\lambda T_F)^{-1}$ represents the gain that is achieved by the use of fault tolerance, under the other assumptions stated. So, for example in Table 5, when $\lambda = 10^{-2}$ and $L = 1$ the gain is approximately 10^6.

5.2 Exact Formulation for Evaluating $Pr(W{<}T_F)$

Unlike the bounding argument used in the last section, our exact derivation of the probability $Pr(W{<}T_F)$ proceeds in two stages, first conditioning on the total number n of faults seen in the lifetime L of the system. It is a well known property of the HPP process [7] that if we condition on the number n of events occurring within a specified time interval and then define $X_1, X_2, \ldots, X_n$ as ordered positions of these n points within that interval, expressed as proportions of its length, then the X_i are (conditionally given n) jointly distributed as the order statistics of an i.i.d. random sample from a uniform distribution on the unit interval $[0, 1]$. This being accepted, we now first fix u with $0{\leq}u{\leq}1$ and ask the question *'What is the probability, P say, that no two of these points are closer than u (conditionally given n)?'*. We can obtain the answer by n-dimensional integration. This is reported in an extended version of this paper available as a technical report [3], which says essentially that P is just the n^{th} power of the total amount of 'slack' remaining within the unit interval after our u-separation constraint is imposed.

This solution conditional given n enables us to complete the exact derivation of the final, unconditional $Pr(W{<}T_F)$ relatively straightforwardly by using the 'chain rule' of conditional probability to 'uncondition on n'. Another fundamental property of the homogeneous Poisson process is that the distribution of n, the count of the number of events occurring within a fixed time interval, upon which the probabilities are conditioned, is Poisson with parameter equal to its mean, which in our case is λL. Then, working for convenience with the 'probability of *no* mishap in a time interval of length L', we have

$$
\begin{aligned}
Pr(W \geq T_F) &= \sum_{n=0}^{\infty} P_{n,\,(T_F/L)} \cdot e^{-\lambda L} \cdot \tfrac{(\lambda L)^n}{n!} \\
&= e^{-\lambda L} \left\{ 1 + \lambda L + \sum_{n=2}^{\lceil \frac{L}{T_F} \rceil} \left(1 - (n-1)\Big(\frac{T_F}{L}\Big)\right)^n \cdot \frac{(\lambda L)^n}{n!} \right\} \\
&= e^{-\lambda L} \left\{ 1 + \lambda L + \sum_{n=2}^{\lceil \frac{L}{T_F} \rceil} \frac{\lambda^n}{n!} \left(L - (n-1) T_F\right)^n \right\} \qquad (14)
\end{aligned}
$$

A few remarks about this exact expression We remark firstly that (14) is essentially a function of just two arguments, λL, λT_F, rather than three (as are the bounds derived in Section 5.1). Thinking now of the function mathematically in these terms, without much concerning ourselves about physical interpretation of the arguments, if we agree to confine ourselves to the ranges $0 < \lambda L < \infty$, and $0 \leq \lambda T_F < \infty$, then we remark that the expression (14) continues to give the correct mathematical Poisson process probability at all points of this domain, including the value of 1 obtained at $\lambda T_F{=}0$. (This is on the understanding that the $\lceil \infty \rceil$ occurring as the upper limit of a sum denotes a sum to infinity in the usual sense of a mathematical limit.) The purpose of stating this last point about the argument domain now to be assumed for this function is related to the practical computation problem associated with (14) which we address briefly in [3]. Note that, apart from this $\lambda T_F{=}0$ case, the expression (14) represents a finite sum throughout the domain identified, although, for certain argument values, the number of terms summed can be astronomically large, which can make a simple-minded numerical computation rather slow. Moreover, some of these awkward parameter ranges may be of real practical interest to us in our application (see end of Section 3).

Note that we can use the common notation for the 'positive part function' h_+, associated with any real-valued function h, to obtain the following slightly different expression, valid throughout the argument domain we have just specified (including $\lambda T_F = 0$).

$$
Pr(W \geq T_F) = e^{-\lambda L} \left\{ 1 + \lambda L + \sum_{n=2}^{\infty} \frac{(\lambda L - (n-1)\lambda T_F)_+^n}{n!} \right\} \qquad (15)
$$

5.3 Some Numerical Results on $Pr(W{<}T_F)$

We decided to test the accuracy of our numerical approximations experimentally, and found that, over the physically realistic parameter ranges of concern to us, the approximations defined are extremely accurate, even at very low order in the Taylor series. This enabled us to produce Figure 2, a contour plot indicating the dependence of the *exact* value of $Pr(W{<}T_F)$ on its two arguments λL, λT_F. The function plotted is, in fact, the *log odds* of $Pr(W{<}T_F)$, chosen in order to ensure that there are some contours near each extreme, $Pr(W<T_F) = 0$ and $Pr(W<T_F) = 1$. In the top right hand corner the contours bunch too closely as the probability of a 'mishap during L' becomes *extremely* close to absolute certainty. (It is difficult to imagine a situation in which the precise values of these large probabilities would be of practical interest.) The rectangular box indicates a subdomain of the arguments over which we have also plotted the *accuracy* of our Taylor series approximation to this exact $Pr(W<T_F)$ function. The technical report[3] contains plots of the percentage inaccuracy that results from the truncation of the approximation after one or two terms.

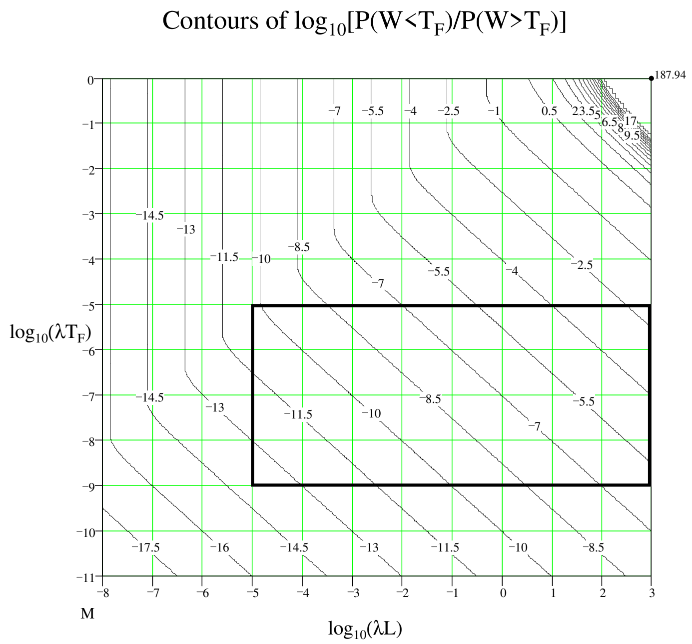

Figure 2: Plots of Exact Value. Notice the log-log scale.

We can illustrate in more detail the interpretation of our numerical results and plots briefly by examining one particular case. Assume $\lambda L = 10^{-2}$ and $\lambda T_F = 10^{-5}$. That is, our system encounters faults with an MTBF of 100 times its lifetime. It is guaranteed to be schedulable provided that it does not, during its lifetime, experience two faults separated by less than one thousandth of the lifetime duration. In such circumstances, we would clearly expect the system to be schedulable with a high probability, P say. This is a log-odds contour plot, so the proximity to the -7 contour indicates that the odds in favour of a system being schedulable with these parameter values are approximately 10^7 to 1. In fact, the bounds on the probability of schedulability, in this situation, obtained by the 'order-of-magnitude' argument of Section 5.1, are 0.4999967×10^{-7} and 1.500477×10^{-7}. The approximations to these bounds, obtained in the two corollaries in Section 5.1, are 0.5×10^{-7} and 1.5×10^{-7}, exactly. The Taylor series approximation allows, in this case, almost arbitrarily accurate calculation of the true value of P with comparatively few terms of the series. In fact we proved that all even-order partial sums, up to the 1000^{th}-order sum, are lower bounds on P, and all odd-order partial sums, up to the 1001^{th}-order sum, are upper bounds. With these particular arguments, the modulus of the fourth order term in the series is less than 10^{-19}, so the sum to only three terms would give an accuracy guaranteed to be better than approximately 11 or 12 significant decimal figures. To eight significant figures, the value of P is $.99948496\times10^{-7}$, corresponding to a log-odds very close to -7 in Figure 2 (at coordinates $(-2,-5)$). The series approximation gives first and second order Taylor approximations of 10^{-7} and $.99948495\times10^{-7}$, respectively. (These numbers are both exact.) See [3] for a detailed derivation of these results concerning high numerical accuracy.

6 Conclusion

We have developed the notion of a probabilistic scheduling guarantee and shown how it can be derived from the stochastic behaviour of fault events. It is reasonable to assume that a fault tolerant system will be designed so as to remain schedulable when dealing with a single fault. The main result of the paper is thus the derivation of a probabilistic guarantee for systems experiencing multiple faults. To do this it has been necessary to formulate a prediction of the likelihood of faults occurring closer together than some specified distance in time. It has also been necessary to use sensitivity analysis to determine the limits to schedulability; that is, the minimum tolerable interval between faults.

Although exact analysis is given for the likelihood of faults occurring quicker than the rate obtained from the sensitivity analysis, perhaps the main result of this paper is a simple derived upper bound for this probability (as given in Corollary 3). A typical outcome of this analysis is that in a system that has a life time of 10 hours with a mean time between transient faults of 1000 hours and a tolerance of faults that do not appear closer than 1/100 of an hour, the probability of missing a deadline is upper bounded by 1.5×10^{-7}. A lower bound is also derived (Corollary 4) and this yields a value of 0.5×10^{-7}. For these parameters the exact analysis produces a value very close to 1.0×10^{-7}.

Interestingly (and perhaps not totally intuitively) the upper, lower and exact formulations for the probabilistic scheduling guarantee all indicate that the threshold value derived from the scheduling and sensitivity analysis has a linear relationship to the probabilistic guarantee. If the threshold value T_F is halved, the probability of missing a deadline is halved. Similarly the length L of execution of the system has a linear impact.

The main obstacle to the use of some of the analysis given in this paper is the lack of empirical data concerning fault arrival times. In the future we aim to address fault clustering and less favourable fault process models. We also aim to move away from the conservative assumption that the system is unschedulable (with probability 1) when faults arrive closer than the threshold value.

References

[1] N. C. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[2] A. Burns, R. I. Davis, and S. Punnekkat. Feasibility analysis of fault-tolerant real-time task sets. *Euromicro Real-Time Systems Workshop*, pages 29–33, June 1996.

[3] A. Burns, S. Punnekkat, L. Strigini, and D.R. Wright. Probabilistic scheduling guarantees for fault-tolerant real-time systems. Technical Report YCS.311, Department of Computer Science, University of York, 1998.

[4] A. Burns and A. J. Wellings. Engineering a hard real-time system: From theory to practice. *Software-Practice and Experience*, 25(7):705–26, 1995.

[5] A. Campbell, P. McDonald, and K. Ray. Single event upset rates in space. *IEEE Transactions on Nuclear Science*, 39(6):1828–1835, December 1992.

[6] X. Castillo, S.P. McConnel, and D.P. Siewiorek. Derivation and Calibration of a Transient Error Reliability Model. *IEEE Transactions on Computers*, 31(7):658–671, July 1982.

[7] D.R.Cox and P.A.W. Lewis. *Statistical Analysis of Series of Events*. Methuen's Monographs on Applied Probability and Statistics, London, 1966.

[8] C. Fetzer and F. Cristian. Fail-awareness: An approach to construct fail-safe applications. In *Proceedings of the 27th Int. Conf. on Fault-Tolerant Computer Systems (FTCS)*, pages 282–291, 1997.

[9] C.-J. Hou and K. G. Shin. Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems. *IEEE Transactions on Computers*, 46(12):1338–1356, 1997.

[10] M. Joseph and P. Pandya. Finding response times in a real-time system. *BCS Computer Journal*, 29(5):390–395, 1986.

[11] D.I. Katcher, H. Arakawa, and J.K. Strosnider. Engineering and analysis of fixed priority schedulers. *IEEE Trans. Softw. Eng.*, 19, 1993.

[12] H. Kim, A.L.White, and K. G.Shin. Reliability modeling of hard real-time systems. In *Proceedings 28th Int. Symp. on Fault-Tolerant Computing (FTCS-28)*, pages 304–313. IEEE Computer Society Press, 1998.

[13] M. H. Klein, T. A. Ralya, B. Pollak, R. Obenza, and M. G. Harbour. *A Practitioner's Handbook for Real-Time Analysis: A Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.

[14] J.P. Lehoczky, L. Sha, and V. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. Tech report, Department of Statistics, Carnegie-Mellon, 1987.

[15] J.Y.T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation (Netherlands)*, 2(4):237–250, 1982.

[16] S. Punnekkat. *Schedulability Analysis for Fault Tolerant Real-time Systems*. PhD thesis, Dept. Computer Science, University of York, June 1997.

[17] S. Punnekkat, R. Davis, and A. Burns. Sensitivity analysis of real-time task sets. In *Proceedings of the Conference of Advances in Computing Science - ASIAN '97*, pages 72–82. Springer, 1997.

[18] K. G. Shin, M. Krishna, and Y. H. Lee. A unified method for evaluating real-time computer controllers its application. *IEEE Transactions on Automatic Control*, 30:357–366, 1985.

[19] S. Vestal. Fixed Priority Sensitivity Analysis for Linear Compute Time Models. *IEEE Transactions on Software Engineering*, 20(4):308–317, April 1994.