Simulation-Based Analysis to Derive Fault Resilience in Real-Time Systems

Flávia Maristela Santos Nascimento[1], **George Lima**[1], **Verônica Cadena Lima**[2]

[1]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
40170-110 – Salvador, BA – Brazil

[2]Departamento de Estatística – Universidade Federal da Bahia (UFBA)
40170-110 – Salvador, BA – Brazil

{flaviasn,gmlima,cadena}@ufba.br

Abstract. *This is an on-going work that aims at deriving metrics that represent fault resilience in real-time systems. Since the described approach is independent of the scheduling policy being considered, it can be used for comparing different scheduling approaches and/or different fault-tolerant real-time systems.*

1. Introduction

There has been intensive research on the field of scheduling for fault-tolerant real-time systems in uniprocessor architectures. They are usually based on time redundancy, that is, upon the detection of an error, a recovery task is scheduled to provide the actions necessary to keep the system correctness. This redundancy model is particularly good for dealing with transient faults. As this kind of faults represent the most frequent one [Ghosh et al. 1998] such a model has been extensively used.

Both the system recovery and the assumed fault models play an important role in the design of the scheduling mechanisms for real-time systems. For example, schedulability analysis has been derived for fixed-priority real-time systems assuming that there is a minimum time between two consecutive errors [Burns et al. 1996, Lima and Burns 2003]. Other authors assume a maximum number of errors during the execution of the system tasks in EDF scheduled systems [Liberato et al. 2000, Aydin 2007]. Nevertheless, despite the large number of such approaches, to the best of our knowledge, there is no systematic way of comparing them.[1] We address this issue here by presenting an innovative approach that aims at establishing comparison metrics able to express the fault resilience of a system independently of the system model considered. The described simulation-based analysis can be used to subsidize the system designer decisions when choosing the fault tolerant mechanisms that best suit their systems.

Some simulation-based analyses have been developed [Wall et al. 2003, Huselius et al. 2007] but none of them has addressed fault tolerance. In the context of fault tolerance, response time analysis has been extended so that it is possible to provide probabilistic guarantees for specific fault models [Burns et al. 1999, Burns et al. 2003, Broster and Burns 2004] but only fixed-priority scheduled systems are considered. Unlike these results, our approach is to be more general and not specific for a given scheduling policy.

[1]For the sake of space we do not cite other related scheduling mechanisms here.

2. Simulation-Based Analysis

The scheme of our analysis method, illustrated in Figure 1, is based on modeling a fault tolerant real-time system. The system execution is represented by two main components, the *scheduler* and the *fault generator*. The former follows a given scheduling policy while the goal of the latter is to generate faults so that task deadlines are missed. Thus, the fault generator acts as an adversary. Both the scheduler and the fault generator take simulation scenarios as input. These scenarios are generated by the *scenario generator* and they represent a set of execution samples that correspond to possible execution scenarios that take place when the actual system is executed.

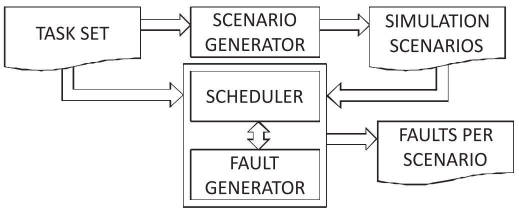

Figure 1. Simulation Environment

Considering a set of m tasks, a simulation scenario j is described by two tuples, $R^j = (r_1^j, r_2^j, \ldots, r_m^j)$ and $C^j = (c_1^j, c_2^j, \ldots, c_m^j)$, where r_k^j and c_k^j represent the release time and the execution time of job k in scenario j, which are considered during the simulation, respectively. The simulation time is dependent on the task being analyzed. For example, in order to analyze a task i in scenario j, it is enough to simulate the execution of the system within $I_i^j = [r, r_i^j + D_i)$, where $r = \min_{k=1,\ldots,m}(r_k^j)$, since this time window is used by the fault generator to generate faults that will make task i miss its deadline. Obviously, in order to analyze a given task i whose execution is subject to faults and interference of other jobs, it is necessary to consider a representative number of execution scenarios. The faults generated may affect task i or any other task that interfere in its execution. Several policies for the fault generator are possible. We are currently considering a conservative approach that minimizes the number of generated faults. This is achieved by generating faults in the active jobs in the time interval I_i^j that produces the highest interference in the analyzed task i.

Consider that task i is being analyzed and that there are N possible execution scenarios. First, assume that N is not too large so that all these scenarios could be generated by the scenario generator. Thus, the simulation-based analysis basically produces the output f_i^j, which represents the number of faults necessary to make task i miss its deadline in scenario $j = 1, \ldots, N$. Based on these values, it is possible to define several metrics that represent the effort made by the fault generator. For example, one may be interested in estimating the minimum or the average effort:

$$E_i^{min} = \min_{j=1\ldots N} \frac{f_i^j}{|I_i^j|} \qquad\qquad E_i^{avg} = \frac{1}{N} \sum_{j=1\ldots N} \frac{f_i^j}{|I_i^j|} \tag{1}$$

Nevertheless, for actual systems the number N may be too large and so it is not computationally viable to consider all possible scenarios. Thus, it is necessary to carry out some statistical analysis capable of estimating values for E_i^{min} or E_i^{avg} based on a number of

$n < N$ scenarios. We are currently working on models based on bootstrap statistical analysis [Efron and Tibshirani 1993], which seem suitable for the kind of data generated by our simulation method.

3. Illustrative Example

In this section we illustrate how our proposed approach works. To do that we consider the same fixed-priority system used by other researchers [Burns et al. 1996, Burns et al. 1999]. The task parameters (C_i, T_i) of each task $i = 1, \ldots, 4$ are respectively the following: $(30, 100)$, $(35, 175)$, $(25, 200)$ and $(30, 300)$. These four tasks are independent and their relative deadlines are equal to their respective periods. The scheduling policy is Rate Monotonic. The recovery model is based on re-execution of the faulty task.

For this example, since tasks are strictly periodic, it is not difficult to see that it is possible to compute the number of possible scenarios. Submitting all 63 found scenarios for the simulator and analyzing the simulator output regarding the lowest priority task, the values of f_4^j, $j = 1, \ldots, 63$ can be found.

It is interesting to compare the results of the proposed approach with the analysis by Punnekkat *et al.* [Burns et al. 1996]. For example, it can be seen that $\min_{j=1,\ldots,63}(f_4^j) = 2$, which coincides with the analysis by Punnekkat *et al.*. However, as can be seen in Table 1, the distribution of f_4^j, $j = 1, \ldots, 63$, gives much more information about the system fault resilience, which can be used to compute the desired metric, e.g. by equation (1). Indeed, the proposed analysis also gives how likely the fault generator can cause missed deadlines with this minimum number of faults. As can be seen from the table, this probability is approximately 0.063. Moreover, notice that the probability that task 4 can cope with at least three generated faults is about 0.625.

	f_i^j						
	2	3	4	5	6	7	Total
Freq.	4	20	11	13	10	6	63
Prob.	0.063	0.297	0.172	0.203	0.156	0.094	1

Table 1. Probability distribution of $f_4^j, j = 1, \ldots, 63$.

It is worth noticing that the probability distribution one may derive from data such as the ones shown in Table 1 does not reflect the probability of tasks missing their deadlines. Indeed, such values are dependent on the fault distribution model for the given system and the fault generator does not take this into account. If one is interested in deriving the probability of having missed deadlines, the fault generator must model fault occurrences in the actual systems (e.g. Poisson distribution), which is beyond the scope of this work.

In conclusion, the simulation-based approach outlined in this paper is expected to be more expressive than the usual scheduling analysis based on worst-case scenarios. A prototype of the simulator is already implemented for some scheduling policies. We are currently working on bootstrap statistical analysis in order to deal with more complex real-time task sets.

References

Aydin, H. (2007). Exact fault-sensitive feasibility analysis of real-time tasks. *IEEE Transactions on Computers*, 56(10):1372 – 1386.

Broster, I. and Burns, A. (2004). Random arrivals in fixed priority analysis. In *1st International Workshop on Probabilistic Analysis Techniques for Real-time and Embedded Systems (PARTES '2004)*, Pisa, Italy.

Burns, A., Bernat, G., and Broster, I. (2003). A probabilistic framework for schedulability analysis. *Embedded Software*, pages 1 – 15.

Burns, A., Davis, R., and Punnekkat, S. (1996). Feasibility analysis of fault tolerant real-time task sets. In *Euromicro Real-Time Systems Workshop*, pages 29–33, L'Aquila, Italy. IEEE Society Press.

Burns, A., Punnekkat, S., Strigini, L., and Wright, D. R. (1999). Probabilistic scheduling guarantees for fault-tolerant real-time systems. In *DCCA '99: Proceedings of the conference on Dependable Computing for Critical Applications*, page 361, Washington, DC, USA. IEEE Computer Society.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Ghosh, S., Melhem, R. G., Mossé, D., and Sarma, J. S. (1998). Fault-tolerant rate-monotonic scheduling. *Real-Time Systems*, 15(2):149–181.

Huselius, J., Kraft, J., Hansson, H., and Punnekkat, S. (2007). Evaluating the quality of models extracted from embedded real-time software. In *ECBS '07: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 577–585, Washington, DC, USA. IEEE Computer Society.

Liberato, F., Melhem, R., and Mossé, D. (2000). Tolerance to multiple transient faults for aperiodic tasks in hard real-time systems. *IEEE Transactions on Computers*, 49(9):906–914.

Lima, G. and Burns, A. (2003). An optimal fixed-priority assignment algorithm for supporting fault-tolerant hard real-time systems. *IEEE Transactions on Computers*, 52(10):1332–1346.

Wall, A., Andersson, J., and Norström, C. (2003). Probabilistic simulation-based analysis of complex real-time systems. *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2003*, pages 257–266.