Debugging Time Behavior of Probabilistic Real-Time Systems

Joaquín Entrialgo, Javier García, José Luis Díaz, Daniel F. García
University of Oviedo
{joaquin, javier, jdiaz, daniel}@atc.uniovi.es

Abstract

In order to detect bugs in the temporal behavior of real-time systems (i.e., discrepancies between the theoretical and the actual behavior), measurements of the running system can be taken. These measurements must then be compared with the specifications and the model used to analyze the feasibility of system. This comparison, which is straightforward when a deterministic model is used, becomes more difficult when probabilistic constraints and models are used. In this paper we discuss this problem and explore some solutions. We propose two heuristic metrics, optimism and pessimism, and show, by means of a case study, how they help in finding bugs.

1. Introduction

In order to prove that a real-time system fulfills all of its timing constraints, various methods have been proposed. Traditional techniques, such as the processor utilization analysis [1, 2] and response time analysis [3], use a model of the tasks of the system where the execution time of the tasks is represented by the worst case execution time (WCET). Using this value, these analyses can obtain an upper bound for the response time of the tasks. Thus, it is possible to determine if all of the tasks will fulfill their deadlines even in the worst circumstances. However, in modern systems, the great variability of the execution times results in too pessimistic WCETs, which leads to oversized systems.

To overcome this problem, another set of techniques, which model the execution time as a random variable, were developed [4, 5, 6]. These probabilistic techniques do not guarantee that all the deadlines are met; instead, they compute the probability of meeting them. The system is deemed feasible if this probability is high enough. This significantly decreases the pessimism of the analysis.

However, when analyzing a real-time system the results of the analysis must be pessimistic even in the probabilistic case, i.e., the probability of missing a deadline in reality must be less than the probability obtained by the analysis technique. This way the analysis is safe. However, some practical problems may arise: the assumptions made by the model might be false, or the techniques can be applied incorrectly. In addition, the actual implementation of the system could have bugs which make the temporal behavior of the real system different from the modeled one.

Measuring the final system and comparing it with the results of the analysis model can reveal these problems. Although measuring cannot be used to prove the correctness of the system, it can bring out bugs or, in absence of them, increase the confidence in the system.

We have developed tools for measuring real-time systems and for comparing the measurements with a traditional (non-stochastic) model of the system [7]. In this paper we explore how this can be carried out when a stochastic model of the system is used in the analysis. The paper is organized as follows: first, the system model is presented; then, the problem of comparing measured values and model distributions is exposed; next, some solutions are discussed; and finally, the conclusions summarize the most important points of the paper and enumerates some open issues.

2. System model

We use the system model presented in [8]. The system is composed of a set of N independent periodic tasks $S = \{\tau_1,..., \tau_i,..., \tau_N\}$. Each task, τ_i, is defined by the tuple $(T_i, \Phi_i, C_i, D_i, M_i)$ where T_i is the period of the task, Φ_i is its initial phase, C_i is its execution time, D_i is its deadline and M_i its maximum allowable ratio of deadline misses.

The execution time, C_i, is a discrete random variable with a known probability function (PF), denoted by $f_{C_i}(\cdot)$, where $f_{C_i}(c) = P\{C_i = c\}$, i.e., the probability of the execution time being c. Alternatively, the execution time distribution can also

be specified using its cumulative distribution function (CDF), denoted by $F_{C_i}(\cdot)$, where

$$F_{C_i}(x) = \sum_{c=0}^{x} f_{C_i}(c) .$$

3. Comparing measurements and model values

In the non-stochastic case, measuring the final system can reveal errors [7]. These errors may be that the system does not follow the specifications (i.e., a deadline miss in a hard real-time system), or that the analysis model and reality are different, for instance an execution time is in fact greater than the WCET used in the model. We refer to both kinds of errors as timing bugs.

However, trying to apply this idea to stochastic real-time systems is not straightforward. Lets consider the system in Table 1 in order to present the problems.

Task	T_i (=D_i)	Φ_i	M_i	C_i	$P(C_i)$
T0	100	0	0.1	10	1
T1	200	0	0.1	20	0.8
				100	0.2
T2	300	0	0.1	30	0.1
				50	0.9
T3	400	0	0.1	10	0.1
				30	0.5
				90	0.39
				300	0.01

Table 1. System example. All times are in ms

Let us assume that the system is measured for two seconds and that in two of the ten releases of task T1 there were deadline misses. This means that the measured deadline miss ratio is 0.2, higher than the maximum allowable ratio M_1, which is 0.1. However, this does not imply an error. There is a certain probability of this happening even when there is no error in the system. If the probability of missing a deadline were exactly $M_1 = 0.1$, then the probability of observing two deadline misses in ten releases can be calculated (assuming independence between deadline misses) with a binomial distribution:

$$p(x; n, p) = \frac{n!}{(n-x)!\,x!} p^x (1-p)^{n-x} \qquad (1)$$

where x is the number of "successes" (deadline misses in our case, 2), n the number of experiments (10), and p the probability of "success" (0.1). Applying this formula the probability obtained is 0.19371024. This means that, even with a valid system, in our experiment there is an almost 20 % of chance of obtaining a deadline miss ratio double that specified.

This example has shown one of the problems of applying measurement to find errors in a system where a certain deadline miss probability is accepted. This problem arises from the fact that the measurements are just a sample of reality.

Another way of finding errors in a traditional (non-stochastic) system is comparing, for all the tasks, the measured execution times with the execution times of the model. If a task presents an execution time higher than specified, the analysis is not valid, because is not pessimistic enough, and must be redone. The power of this comparison is that it can find potentially unfeasible systems even when no deadline miss is observed.

Applying this to stochastic real-time systems is not straightforward either. In deterministic systems, the operation for determining if the model is pessimistic enough consists only of comparing two numbers for each task: the WCET in the model against the measured WCET. In stochastic systems, what must be compared are two probability distributions: the distribution of execution time of the model and the distribution (profile) of the measured execution time.

For example, in another experiment with the system presented in Table 1 we might measure that nine of the ten releases of task T1 had an execution time of 10 ms, and the remaining release had an execution time of 105 ms. Is this measured distribution worse or better than the model distribution?

Obviously, the observed execution profile does not follow exactly the PF of the model, but this perfect fit would be impossible, due to the low number of observations. Even with a big number of observations, both functions are not required to be equal because the model is allowed to be approximated, as long as this approximation is safe, i.e., pessimistic, as defined in [8]. So the kind of comparison to be performed is one which can tell us if the distribution assumed in the model is more pessimistic or not than the one derived from the measurement. The one-sided Kolmogorov-Smirnov goodness-of-fit test carries out this kind of comparison for continuous and independent random variables. However, we have initially discarded this test because execution profiles are not continuous and, in many systems, independence cannot be assumed.

In [8], the pessimism (or "worse than" relationship among random variables) is formalized as the first-order stochastic dominance. In this work, a random variable X is said to be "worse than" another random variable Y if $F_X(x) \leq F_Y(x)$ for all x. Graphically this means that the CDF of X is always below the CDF of Y.

This suggests that we could simply check whether the measured CDF is always above the CDF derived

from the model and the analysis. But again this kind of test would not be valid. Fig. 1 depicts the CDF of the measured computation time, and the one derived from the analysis. It can be seen that both curves cross. Since none of the curves is always above the other, the relationship "worse than", as defined in [8] does not apply.

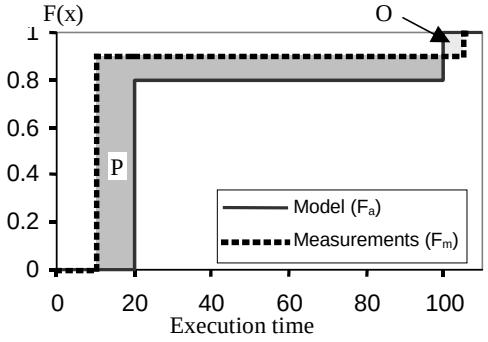

Fig. 1. CDF of model and measurements for task T1

It can be seen in Fig. 1, that in the aforementioned example there is more probability of obtaining an execution time shorter than or equal to 10 ms in the measurements than in the model. This means that for that point the model distribution is more pessimistic than the measurements, i.e., the model gives more probability for longer execution times. However, in 100 ms this trend changes: there is a greater probability of obtaining an execution time of 100 ms in the model than in the measurements; i.e., for this value, the model is optimistic. Then it is not straightforward to draw a conclusion about the validity of the model.

The random nature of the system and the measurements makes an exact comparison unfeasible, but if models are pessimistic, measurement CDFs are expected to be over the model CDFs "most of the time". We will use this idea to define a heuristic to compare the model and the measurements.

4. Metrics for comparison

In order to overcome some of the problems presented in the previous section, we propose using two heuristic metrics which will be called *optimism* and *pessimism*. This metrics can be used to find bugs in the system (undesired behavior and differences with the model) and to discover in which task the bug is found.

Both metrics are defined comparing the execution time CDFs of the model and the measurements.

Let F_m be the CDF of the measurements, F_a be the CDF of the analysis model, $X_O=\{x|F_a(x)>F_m(x)\}$, $X_P=\{x|F_a(x)<F_m(x)\}$ and $x_{max}=\min\{x|F_m(x)=1$ and $F_a(x)=1\}$. Without loss of generality we assume that all the execution times are scaled so they are integers.

Definition 1. The *optimism* is defined as

$$O = \frac{1}{x_{max}} \sum_{x \in X_O} (F_a(x) - F_m(x)) \qquad (2)$$

Definition 2. The *pessimism* is defined as

$$P = \frac{1}{x_{max}} \sum_{x \in X_P} (F_m(x) - F_a(x)) \qquad (3)$$

Pessimism is defined as the area between the model CDF and the measurements CDF when the model CDF is under the measurements CDF. Optimism is the area between the model CDF and the measurements CDF when the model CDF is over the measurements CDF. Both quantities are divided by x_{max} so that their range is [0, 1].

In Fig. 1 optimism is marked with an *O* and pessimism with a *P*. When pessimism is positive, and optimism is zero, this means that the model CDF is always below the measurements CDF. This is good, because this means that the model provides pessimistic estimations. Analogously, having low pessimism and high optimism in the measurement is bad, because this means that probably the model is being optimistic.

In order to show the utility of these metrics, a case study is now presented. A system with the parameters shown in Table 1 has been implemented in a Pentium III at 800 MHz running QNX 6.1.

Table 1 represents the parameters of the "real" system. These parameters are actually unknown to the analyst. The analysis will use instead an approximated model of the system. For the analysis being valid, this approximation has to be pessimistic. In order to simulate this pessimism, we will use as system model the same parameters of Table 1, but increasing the execution times by 10%. With this model, we have carried out a probabilistic response time analysis (using [5]), which gives the results presented in Table 2. The CDF resulting from this analysis will be compared against the measurements of the real system (whose parameters are those of Table 1). It is to be expected that the analysis model will show pessimism when compared to the measurements.

Task	Deadline miss probability	Deadlines misses forecasted in 60 seconds
T0	0	0
T1	0	0
T2	0	0
T3	0.0312	4.68

Table 2. Case study analysis

In order to test the ability of the metrics to find bugs, an error has been introduced in the implementation of the system. The two execution

times of task T1 are the consequence of a conditional sentence with two paths. The condition has been reversed and, so, the probabilities of the two execution times of task T1 have been interchanged. Therefore, in the implemented system there is a 0.2 probability of having an execution time of 20 ms and an 0.8 probability of having an execution time of 100 ms.

The system was measured for 60 seconds using the tools presented in [7] and the pessimism and the optimism have been computed using Eq. (2) and (3). Table 3 presents the metrics for each task, with pessimism and optimism given in percentage.

Task	Pessimism	Optimism	Missed deadlines
T0	9.18 %	0 %	0
T1	2.11 %	43.00 %	0
T2	8.14 %	0.15 %	0
T3	1.11 %	2.51 %	15

Table 3. Case study metrics

From these results, the debugging tool can check if the number of missed deadlines for each task is "reasonable" or not, according to the model. For example, task T3 had 15 deadline misses. According to the model, the probability of deadline misses of task T3 is 0.0312. From this figure and using the binomial cumulative probability function, we can compute the probability of having 15 or more deadline misses in 60 seconds of execution (which represents a total of 150 releases of the task). This probability is $7.92 \cdot 10^{-5}$, which is extremely low. So the debugging tool can pinpoint that there is a problem with the response times of task T3.

Without the metrics, the first place where an analyst would look for the error would be in task T3. However, a problem in the response time of task T3 does not necessarily imply a problem in the implementation of task T3.

Using the metrics in Table 3, it can be seen that there is much more optimism in task T1 than in task T3, which (correctly) indicates that the bug is in task T1.

5. Conclusions

Although measurement can not prove the correctness of a real-time system, measuring is a way to detect bugs and increase the confidence in the system. Measurement provides more valuable information when its results are compared with a model of the system and not just with the specifications. This comparison is easier to carry out when a traditional (non-probabilistic) analysis technique is used than with a probabilistic technique.

In this paper we have described some of the problems that arise in measuring and detecting timing bugs with probabilistic techniques. The first problem is that constraints are probabilistic and, therefore, finding a deadline miss does not necessarily mean a bug. The second problem is that in order to determine if the execution time distribution of the model is not as pessimistic as the real distribution, both distributions must be compared and this comparison is not straightforward.

Two metrics have been proposed to detect bugs in the system: optimism and pessimism. These metrics are based on comparing the CDFs of the execution time distributions. A case study has been shown in order to demonstrate its applicability.

The problems presented here are an open field in which more work must be carried out. This includes, for example, finding new metrics, developing a set of problems in order to compare different metrics, relating the measurement length with the reliability of the metrics, and assessing the use of hypothesis tests as the Kolmogorov-Smirnov test to find errors.

6. References

[1] L. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of ACM*, 20(1):46–61, 1973.

[2] J. P. Lehoczky. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proc. of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, Dec. 1990.

[3] K. Tindell, A. Burns, and A. J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Real-Time Systems*, 6:133–151, 1994.

[4] G. Bernat, A. Colin, and S. Petters. WCET Analysis of Probabilistic Hard Real-Time Systems. In *Proc. of* the 23rd IEEE Real-Time Systems Symposium, Dec. 2002.

[5] J. L. Díaz, D. F. García, K. Kim, C.-G. Lee, L. L. Bello, J. M. López, S. L. Min, and O. Mirabella. Stochastic Analysis of Periodic Real-Time Systems in a Real-Time System. In *Proc. of the 23rd IEEE Real-Time Systems Symposium*, pages 289–300, Austin, Texas, Dec. 2002.

[6] M. K. Gardner. *Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems*. PhD thesis, University of Illinois, Urbana-Champaign, 1999.

[7] J. Entrialgo, J. García and D.F. García. Measurement-based Analysis of Real-time POSIX Applications. In *Proc. of the 7th IASTED International Conference on Software Engineering and Applications*, Nov. 2003.

[8] J.L. Díaz and J.M. López. Safe Extensions to the Stochastic Analysis of Real-Time Systems. Technical Report, Departamento de Informática, University of Oviedo, 2004. Also available at http://www.atc.uniovi.es/research/SESARTS04.pdf