Challenges and New Trends in Probabilistic Timing Analysis

S. Quinton† and R. Ernst† D. Bertrand* and P. Meumeu Yomsi*

†IDA - TU Braunschweig; Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
Email: {quinton, ernst}@ida.ing.tu-bs.de

*EPI TRIO - INRIA Nancy Grand-Est; 615, rue du Jardin Botanique, 54000 Villiers-lès-Nancy, France
Email: {dominique.bertrand, patrick.meumeu}@inria.fr

***Abstract*—Modeling and analysis of timing information are essential to the design of real-time systems. In this domain, research related to probabilistic analysis is motivated by the desire to refine results obtained using worst-case analysis for systems in which the worst-case scenario is not the only relevant one, such as soft real-time systems. This paper presents an overview of the existing solutions for probabilistic timing analysis, focusing on challenges they have to face. We discuss in particular two new trends toward Probabilistic Real-Time Calculus and Typical-Case Analysis which rise to some of these challenges.**

***Index Terms*—Real-time systems, Stochastic analysis, Probabilistic Real-Time Calculus, Typical-Case Analysis.**

I. Introduction

Timing modeling and formal verification of real-time systems are currently hot topics in embedded systems research, e.g., within the TIMMO-2-USE project [2]. Real-time systems very often consist of multiple resources such as processors or buses upon which multiple tasks are executing concurrently. To arbitrate between tasks trying to execute at the same time, different scheduling policies are possible, which have different consequences on the runtime behavior of tasks. Furthermore, tasks may activate each other and exchange data. These complex interferences are very hard to capture by simulation, thus making formal analysis essential in order to obtain safe information on the behavior of the system.

Two compositional approaches have successfully tackled this problem, namely Real-Time Calculus (RTC) [5] and Compositional Performance Analysis (CPA) [11]. These methods are compositional in the sense that they use *local* information about tasks and resources (such as execution times or scheduling policies), which can be determined without taking into account any interference with other tasks or resources, to come up with *global* information about them, that is, information about how tasks behave *within* a given system (e.g. in terms of response times). These methods are efficient and safe, however their results are often rather pessimistic when compared to simulations of the system that is under analysis. This discrepancy has two explanations.

- ▷ First, the result of the analysis is an overestimation of the actual worst-case scenario possible at runtime.
- ▷ Second, the actual worst-case scenario may be very rare and thus may never occur during the simulations of the system, although it is indeed possible.

978-3-9810801-8-6/DATE12/©2012 EDAA

Probabilistic approaches (often called *stochastic*) have been developed in order to refine results in the latter case. The basic principle of probabilistic timing analysis is to assume that some local probabilistic information is available (e.g., describing the execution times of tasks) and use it in order to come up with some global probabilistic information (e.g., on the response times of tasks or on the end-to-end latencies). Initial solutions [7] focused on the monoprocessor case where several tasks execute on a single resource. In this paper we discuss current trends toward multiprocessor analysis. Note that a probabilistic analysis, just like any other analysis, remains useless unless it satisfies the following conditions.

1) It is efficient enough to scale to the size of real systems.
2) It provides results that are meaningful and helpful for the system designer.
3) The model used for the analysis is practical — that is, either it is simple enough to be provided by the system designer, or it is automatically derivable, e.g. from a trace.
4) Any assumption made by the analysis is formally described and can be validated either by the designer or by some automated method (e.g., using statistical tests).

In the context of probabilistic timing analysis, these four conditions represent four challenges that are at the core of current research in the domain.

Efficiency is indeed a key issue as initial approaches aiming at exact results were in practice computationally too expensive. The idea of reducing the complexity by using *pessimism* on some parameters was introduced in [8]: the distributions are then interpreted as a safe approximation of the system's behavior rather than as an exact representation. In such a pessimistic approach several optimizations are possible in terms of efficiency. For example, there is currently active research focusing on *re-sampling* techniques [25], where the goal is to reduce the co-domain of distributions while preserving soundness of results.

Unfortunately, while reducing computation costs, pessimism also degrades the quality of results. This means that efficiency

cannot be the only criterion to evaluate an analysis method and the accuracy of results is also essential to ascertain that these results are meaningful. In particular, one should keep in mind that probabilistic analysis must be more precise than worst-case analysis to be useful. Besides, not all representations of probabilistic information are equally expressive. Most existing probabilistic approaches focus on (possibly pessimistic) distributions on response times and thus cannot provide any information on the behavior of the system (e.g., regarding bursts of missed deadlines) within a bounded time window. Therefore one must pay attention to which type of probabilistic information is actually needed or useful for the design of a given system. *Probabilistic Real-Time Calculus* (PRTC) [26] and *Typical-Case Analysis* [23] are emerging trends which can provide information about bursts.

Regarding the third above-mentioned challenge, namely how to obtain the model on which probabilistic analysis is performed, one must admit that designing a probabilistic model manually is not always an easy task. For this reason it is essential to provide methods for deriving such models from traces of system execution or simulation and then describe what information can be reasonably extracted from such a trace. Typically, one will assume that local information is reliable while global information is not and must be obtained by formal analysis. Furthermore, randomness properties of variables must be validated using standard tests on traces, wherever randomness is assumed [14].

Finally, the most common assumption made by existing approaches (including PRTC) is the *independence* of the various random variables used to represent local probabilistic information. This typically implies that the execution time of any instance of a task is correlated neither with the execution time of any other instance of the same task, nor with the execution time of any instance of another task. This assumption is reasonable in different contexts [1] and there exist statistical tests to check that it is realistic. However it was shown in [12] that if there are indeed dependencies in the system then the analysis must take them into account.

Dealing with dependencies is a hard task especially because determining and representing them is very difficult. Moreover, adopting a pessimistic approach such as using *copulas* leads to results that are as pessimistic as the ones produced by worst-case analysis. As a consequence of this situation, a different perspective on the problem is needed. This is what is proposed by *Typical-Case Analysis* (TCA) [23]. Strictly speaking, TCA is not a probabilistic approach because it provides firm guarantees on the behavior of a system within a bounded time window but its outcome is sufficiently similar to that of a probabilistic analysis to be of interest to the same research community and industrial practitioners. For example, TCA can prove that the response time of a task cannot be larger than a given value more than m times out of k consecutive executions. This type of properties is related to *weakly-hard* constraints and (m, k)-firm systems, as explained in Section IV.

Organization of the paper

This paper is organized as follows. Section II describes the initial monoprocessor approaches developed mainly by Díaz et al. over the past decade. Section III presents Probabilistic Real-Time Calculus as a new approach for providing precise results when randomness and independence can be assumed. Section IV describes Typical-Case Analysis which focuses on systems where this hypothesis does not hold. Finally Section V discusses challenges that still remain ahead of us.

II. Stochastic analysis

In this section we present the state of the art in the domain of monoprocessor probabilistic analysis initiated by [27]. This is performed with an emphasis on the work developed by Díaz et al. for improving the efficiency of the method, which they call *stochastic analysis*. Stochastic analysis aims at providing a distribution of the response times of each task in a system when the distributions of the execution times of tasks are given — whereas in classical worst-case analysis only the best-case and worst-case execution times (denoted by BCET and WCET, respectively) are known. Section II-B discusses how the execution time distributions are obtained in practice. As each task is assigned a relative *deadline* which defines for each of its instances (i.e., execution of the task) a date by which this instance must be completed, the goal of the analysis is to formally determine "how often" deadlines are missed.

Most existing methods based on probabilistic execution times, except the one presented by Díaz et al. in [7], introduce worst-case assumptions to simplify their analysis. Some typical assumptions are: the critical instant assumption [27], [9], restrictions on the load such as the heavy traffic condition [17], or restrictions on the occurrence of preemptions [21]. For these reasons we focus on the work of Díaz et al. on the analysis of *periodic* tasks as it does not assume any such worst-case or restrictive conditions.

A. Principle of stochastic analysis

The technique presented in [7] is based on two basic assumptions. First, the *offset* of each task (i.e., the time of its first activation) is fixed and known a priori. Second, tasks do not share any resource other than the one on which they execute, such that execution is lockfree. The scheduler is assumed to be *Fixed Priority Preemptive* (FPP) [19]. Note that this technique has been extended in [8], [20] to handle resource sharing, priority allocation and other schedulers such as Earliest Deadline First (EDF). The execution time of each instance $\tau_{i,j}$ of a task τ_i is modeled by a random variable $\mathcal{C}_{i,j}$, and the random variables of all instances of a task are assumed to follow the same distribution. Moreover all random variables are supposed to be independent to simplify the computation of their sums. Under this hypothesis, the sum of two random variables can be obtained by convolution.

Based on these execution time distributions Díaz et al. compute, for each instance $\tau_{i,j}$, the distribution of the random variable $\mathcal{R}_{i,j}$ characterizing its response time. This method is

the adaptation of the deterministic analysis presented in [13] to the stochastic case. $\mathcal{R}_{i,j}$ is obtained as follows.

$$\mathcal{R}_{i,j} \stackrel{\text{def}}{=} \mathcal{B}_{i,j} + \mathcal{C}_{i,j} + \mathcal{I}_{i,j}$$

where $\mathcal{I}_{i,j}$ and $\mathcal{B}_{i,j}$ respectively denote the interference of higher priority tasks and the *backlog*, i.e., the influence of the instances of tasks with a priority higher than or equal to that of τ_i that have not completed yet when $\tau_{i,j}$ is activated. Once the distribution of $\mathcal{R}_{i,j}$ is known, the schedulability of $\tau_{i,j}$ relies on the probability of $\mathcal{R}_{i,j}$ being less than or equal to its deadline.

Since the execution of the system consists of an infinite sequence of instances, it follows that an infinite number of instances should be analyzed. However, it has been proven in [7] that the analysis can be limited to a bounded time interval in two cases. First, when the maximum processor utilization is less than one, it is sufficient to compute the backlog at the end of the first *hyperperiod* (i.e. the least common multiple of the periods of all tasks) and perform the analysis on all instances contained in the second hyperperiod. Second, when the mean processor utilization is less than one, the stochastic process defined as the sequence of backlogs at the beginning of successive hyperperiods can be modeled by using a Markov chain. In addition, it has been shown that a stationary distribution of the backlog, called *steady-state backlog*, exists. Based on this steady-state backlog, all significant probabilities can be computed.

B. Challenges and current trends

In practice, computing the steady-state backlog for the second case described in the previous section is a challenge in terms of efficiency. Except for simple task sets, the only existing method for computing this steady-state backlog is by using a fixpoint algorithm in which one calculates the backlog after each successive hyperperiod. However this method results in a more pessimistic backlog at each new iteration step, so the backlog obtained by a truncated iteration may be optimistic, that is, unsafe. More details on this issue can be found in [8].

Furthermore, even if we use the approximation found after a certain number of iterations, the number of points representing the steady-state backlog might be too large to be stored in the memory. One solution to circumvent this problem consists in truncating the distributions and/or *re-sampling* them. These operations can be achieved in a pessimistic manner as proved in [8]. More precisely, it has been shown that clusters of probabilities can be aggregated towards the worst (i.e. largest) execution time of the cluster. Such transformation clearly is pessimistic and reduces the number of points to store in memory. Note that the complexity of the analysis significantly depends on the number of non-zero points in the distributions. Some experimental results in this direction are presented in [15]. Other studies such as those presented in [25], [22] apply pessimistic re-sampling not only to the input execution time distributions but also during the backlog computation.

A second main challenge for stochastic analysis is to obtain the distributions of execution times on which the model is based. These distributions can be obtained by measurement, or by using hybrid techniques [6]. However it is well known that simulation cannot cover all cases as illustrated in Figure 1: the black full line represents the exact probability distribution of the execution time of a task while the blue dashed line shows the distribution as it could be actually observed and measured. One can notice in particular that the measured WCET (resp. BCET) differs from the exact WCET (resp. BCET), as large execution times with a very low probability may never be observed in a simulation. Besides, the exact WCET is usually smaller than the safe bound for it (denoted safe WCET in the figure) which can be obtained by static analysis [28].

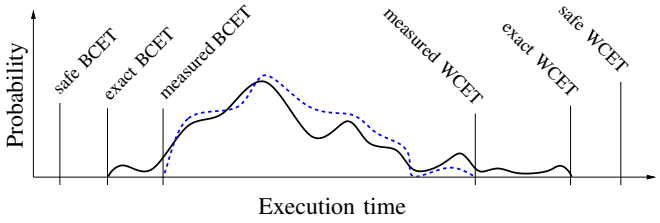

Figure 1. Exact and measured distributions on execution times.

Finally, we have mentioned that existing stochastic approaches rely on the assumption that the execution times of task instances can be represented as independent random variables. If this is not ensured by construction [1], this assumption must be justified, in particular when distributions are derived from traces of system execution or simulation. First, one must check that execution times can indeed by modeled by random variables, then verify that the independence hypothesis cannot be rejected, focusing for example on linear dependencies etc. Note that the best result that can be obtained using statistical testing is that the independence hypothesis *cannot be rejected*, in contrast with a result that would validate it. A detailed case study is presented in [14] where such tests are performed to derive distributions on inter-arrival times from traces (instead of execution times).

III. Probabilistic Real-Time Calculus

Stochastic analysis as presented in the previous section is monoprocessor and restricted to periodic tasks. The technique presented in this section is based on Real-Time Calculus (RTC) [5] which uses abstract and very expressive models for representing activations and execution times (namely *request* and *service* curves). As a result this approach does not suffer from the restrictions imposed by stochastic analysis on the activation pattern of tasks and on the number of resources (processors) in the system. Moreover, it can handle probabilistic information on the inter-arrival time of task activations.

A. Principle of RTC

Real-Time Calculus is based on the notion of *event streams* which describe how tasks activate each other. An event stream is represented by the upper bound $\alpha_u(\Delta)$ and the lower bound $\alpha_\ell(\Delta)$ on the number of event occurrences (e.g. activations)

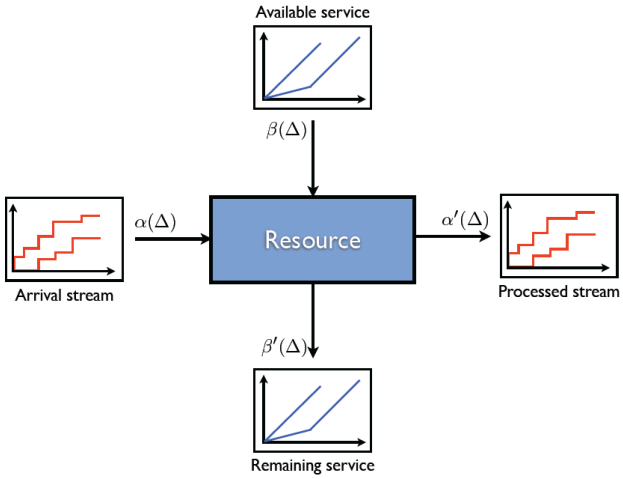

Figure 2. Basic principle of Real-Time Calculus: $\alpha(\Delta)$ and $\beta(\Delta)$ are the inputs whereas $\alpha'(\Delta)$ and $\beta'(\Delta)$ are the outputs.

within any time interval of length Δ. Similarly, the *service* offered by a resource is specified using the upper function $\beta_u(\Delta)$ and the lower function $\beta_\ell(\Delta)$ which describe the maximum and minimum number of serviced (processed) events, respectively, within any interval of length Δ. Hence, if $I_\mathcal{R}(t)$ stands for the number of input events from time 0 up to time t in a resource $\mathcal{R}$, and $C_\mathcal{R}(t)$ stands for its processing capability at time t, it holds that:

$$\begin{cases} \alpha_\ell(\Delta) \leq I_\mathcal{R}(t) - I_\mathcal{R}(t-\Delta) \leq \alpha_u(\Delta) \\ \beta_\ell(\Delta) \leq C_\mathcal{R}(t) - C_\mathcal{R}(t-\Delta) \leq \beta_u(\Delta) \end{cases}$$

Given functions α_u and α_ℓ corresponding to an event stream arriving at a resource $\mathcal{R}$, and the service functions β_u and β_ℓ offered by $\mathcal{R}$, the intuitive idea behind the RTC paradigm (as illustrated by Figure 2) consists in computing the timing properties of the processed stream and the remaining processing capacity, i.e., functions α'_u, α'_ℓ, β'_u and β'_ℓ, as well as the maximum backlog and the delay experienced by the stream in $\mathcal{R}$. These functions are obtained using kernel formulas [16] whose efficient implementation is challenging.

The computed functions α'_u and α'_ℓ are then used as inputs to the next resource on which this stream is further processed. By repeating this procedure until all resources in the system have been considered, timing properties of the fully-processed stream can be determined as well as its end-to-end event delay and the global backlog. This forms the basis for the composition of local analyzes of individual resources to obtain formal timing information about the global system.

B. Principle of PRTC

Just like RTC was inspired by Network Calculus [16], *Probabilistic Real-Time Calculus* (PRTC) [26], [24] was inspired by Probabilistic Network Calculus [29]. The key idea is to refine the classical model of RTC based on four functions α_u, α_ℓ, β_u and β_ℓ, by replacing these four functions with a finite [1] set of functions α^i_u and α^i_ℓ for event streams (with $1 \leq i \leq n_i$ and $n_i \in \mathbb{N}^+$), each of which is bounded from above by α_u and from below by α_ℓ, then a set of functions β^k_u and β^k_ℓ for resource service (with $1 \leq k \leq m_k$ and $m_k \in \mathbb{N}^+$), each of which is bounded from above by β_u and from below by β_ℓ. In addition each function is associated with a parameter $p \in [0,1]$ corresponding to the probability of the actual stream being above (respectively below) the given function in the interval domain. Using convolution, as was already the case for the classical RTC, one can compute the distribution of the (output) probabilistic processed and remaining functions based on the (input) arrival and service functions. Finally, when all streams have been processed, one can perform a schedulability analysis as presented in Section II.

1. Only discrete distributions of arrival and service curves are considered.

C. Challenges and current trends

It is unlikely that system designers will be able to provide directly models for PRTC. Therefore one must explain how such models can be derived from execution or simulation traces. To do so, we consider a probabilistic model similar to that of Section II where not only execution times are probabilistic but also periods. Probabilistic periods model the inter-arrival times of task activations. Note that this model is well suited for describing sporadic activations. We have already discussed how such a model of computation can be obtained from a trace. How probabilistic arrival and service functions can be computed from this model is a rather technical issue and thus beyond the scope of this paper. A detailed explanation is available in [26].

On a different level, despite all the benefits resulting in the usage of PRTC, a challenge of this approach remains its high computational complexity. Indeed, even though PRTC provides useful information about the timing behavior of her system, the efficiency of the method is obviously dependent on that of RTC which is already problematic.

IV. Typical Worst-Case Analysis

Before presenting Typical-Case Analysis [23], let us discuss the specific challenges that it was designed to address. As already mentioned, existing stochastic approaches as well as Probabilistic Real-Time Calculus rely on the assumption that the execution times of tasks (and inter-arrival times for PRTC) can be represented as independent random variables. This is in general not the case and ignoring these dependencies is not safe as shown in [12]. So far there exists no usable solution when the independence hypothesis does not hold. A conservative approach using *copulas* was proposed in [4] and [12] but in practice it yields pessimistic results that are close to those produced by worst-case analysis. As a consequence to this situation, a different perspective on the problem is needed.

It is important to keep in mind that approaches based on distributions do not provide information on the behavior of the system in a bounded time window. For example, it is not possible to distinguish, based only on the distribution of the inter-arrival time of events, the two traces of Figure 3.

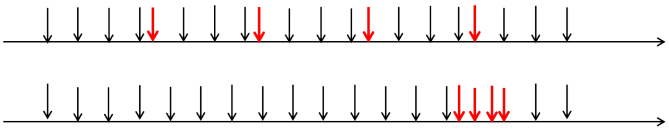

Figure 3. Limits of probabilistic information: in both traces the probability of a small inter-arrival time is of $\frac{1}{4}$.

Typical-Case Analysis (TCA) addresses these two issues. As already mentioned, it is not strictly speaking a probabilistic approach because it provides firm guarantees on the behavior of a system within a bounded time window but its outcome is sufficiently similar to that of a probabilistic analysis to be of interest to the same research community and industrial practitioners. More precisely it comes up with timing information of the form: "*The response time of task τ cannot be larger than R more than m times out of k consecutive executions.*" This type of properties is related to *weakly-hard* constraints [3] which state in a similar way how many deadlines may be missed out of a sequence of consecutive executions. Note however that the outcome of Typical-Case Analysis can be used for other purposes than schedulability analysis, such as getting some quantitative timing information about critical tasks in a mixed-critical system. Furthermore TCA is not restricted to periodic tasks as in [3], but can use the same event stream models as compositional analysis, i.e. arrival curves [5] or PJD (periodic with jitter and minimal distance) models [11]. Let us also mention (m, k)-firm systems [10] which focus on enforcing this type of properties by defining the appropriate scheduling policy rather than analyzing it.

A. Principle of Typical-Case Analysis

Our approach is based on compositional performance analysis (CPA) [11]. We suppose that we are given:

1) a safe (worst-case) model M of the system;
2) an approximate (typical-case) model M' of the system where unlikely behaviors of M are omitted — e.g. an "almost periodic" task in the safe model is considered periodic, or a worst-case execution time that rarely happens in practice is ignored;
3) a distance between both models which accurately describes the approximation made in M'.

We then perform separately two CPAs, one on M and the other one on M'. In a second phase, we use the result of the CPA of M and the distance between M and M' to formally describe the accuracy of the obtained approximate worst-case response time (WCRT) by providing for any k the smallest safe value of m such that the response time of the task under study cannot be larger than the approximate WCRT more often than m times out of k consecutive executions.

Let us illustrate this abstract description of Typical-Case Analysis on the first example of Figure 4, where a task is activated periodically with some additional sporadic overload activations. In this case the difference between the safe and the typical-case models is formally represented using what we call

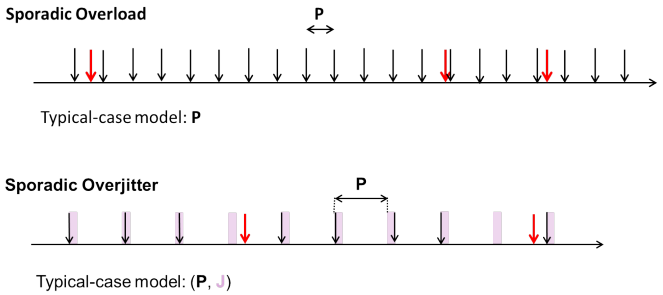

Figure 4. Two activation traces for which Typical-Case Analysis can be efficiently applied, along with their typical-case model.

an *overload model*, which models the trace only consisting of the red activations. An upper bound describing the accuracy of the approximate WCRT can be obtained based on the following observations related to a *busy window* analysis [18] of the task:

- one overload activation is in at most one busy window;
- a busy window contains at most K activations where K is the worst-case number of activations in a busy window;
- in absence of overload activations in a busy window, all response times are smaller than the approximate WCRT.

One can then conclude that one overload activation cannot impact more than K response times. The full description of this example is available in [23].

The same approach can be followed in other cases, e.g. to handle *overjitter* at the input of a task, as illustrated in Figure 4, or to model efficiently execution times of tasks.

B. Challenges and current trends

One of the nice properties of this approach is that it is extremely efficient in terms of computational complexity. As regards usefulness of the result, we have performed experiments with a realistic example where some tasks are activated both periodically (time-triggered) and aperiodically (event-triggered) thus leading to an activation model which is periodic with a sporadic overload. Experiments show that TCA can result in an approximate WCRT which is dramatically smaller than the safe one, still giving guarantees that fit the important class of weakly hard real-time systems. The interested reader is again referred to [23].

Let us now discuss the issue of how to obtain the approximate models on which TCA is based. These models can be directly specified or derived from execution or simulation traces. In the latter case it is assumed, as is the case for derived worst-case analysis, that the trace provides good estimates on the *local* behavior of tasks while it does not have to be trusted with respect to the *global* behavior of the system. More precisely, for every size of an "observation window" we only suppose that the worst-case scenario within such a window appears in the trace, as regards inter-arrival times and execution times.

One then derives automatically a worst-case model of each task and manually defines a typical-case model based on the trace. The distance between safe and approximate models is

obtained by first formally identifying in the trace the "error" events which induce a difference between safe and approximate WCRT and then following the principles of the δ and η functions of [11] to represent how often these events may appear. Again we only suppose that the worst-case scenario with respect to error events appears in the trace for every observation window.

V. Conclusion

We have presented an overview of the state of the art in probabilistic timing analysis with a focus on the main challenges that must be addressed, namely: efficiency of the computation, usefulness of the results to the system designer, possibility to obtain in practice the model on which the analysis is based and to verify that the assumptions made do indeed hold. We described in more detail emerging trends that propose new answers to these problems including an alternative model and analysis that can be applied where randomness and stochastic independence of system variables cannot be assumed or where the system behavior in a bounded time window is of interest.

As should now be clear to the reader, more research is needed in the domain of probabilistic timing analysis. However the recent results obtained by Probabilistic Real-Time Calculus (PRTC) and Typical-Case Analysis (TCA) show that current limitations on the applicability of existing approaches could be soon overcome. To conclude, let us note that an interesting development in the future would be to combine PRTC and TCA so as to benefit from the precision of PRTC for random variables which are independent and still be able to handle the remaining interdependent and non-random variables.

Acknowledgment

This work was funded by the ITEA2 project TIMMO-2-USE (EUREKA cluster N° 3674) through the French Ministry for Industry and Finances and the German Ministry of Education and Research (BMBF) under the funding ID 01IS10034. The responsibility for the content rests with the authors.

References

[1] PROARTIS: PRObabilistically Analysable Real-TIme Systems. [Online]. Available: http://www.proartis-project.eu/

[2] TIMMO-2-USE: TIMing MOdel - TOols, algorithms, languages, methodology, and USE cases. [Online]. Available: http://timmo-2-use.org/

[3] G. Bernat, A. Burns, and A. Llamosí, "Weakly hard real-time systems," *IEEE Trans. Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[4] G. Bernat, A. Burns, and M. Newby, "Probabilistic timing analysis: An approach using copulas," *J. Embedded Computing*, vol. 1, no. 2, pp. 179–194, 2005.

[5] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proceedings of DATE'03*. IEEE Computer Society, 2003, pp. 190–195.

[6] L. David and I. Puaut, "Static determination of probabilistic execution times," in *Proceedings of ECRTS'04*. IEEE Computer Society, 2004, pp. 223 – 230.

[7] J. Díaz, D. García, K. Kim, C.-G. Lee, L. Lo Bello, J. López, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Proceedings of RTSS'02*. IEEE Computer Society, 2002, pp. 289–300.

[8] J. Díaz, J. López, M. García, A. Campos, K. Kim, and L. Bello, "Pessimism in the stochastic analysis of real-time systems: concept and applications," in *Proceedings of RTSS'04*. IEEE Computer Society, 2004, pp. 197 – 207.

[9] M. K. Gardner and J. W.-S. Liu, "Analyzing stochastic fixed-priority real-time systems," in *Proceedings of TACAS'99*, ser. LNCS, vol. 1579. Springer, 1999, pp. 44–58.

[10] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignement technique for streams with (m, k)-firm deadlines," *IEEE Trans. Computers*, vol. 44, no. 12, pp. 1443–1451, 1995.

[11] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the SymTA/S approach," in *IEE Proceedings Computers and Digital Techniques*, 2005.

[12] M. Ivers and R. Ernst, "Probabilistic network loads with dependencies and the effect on queue sojourn times," in *Proceedings of QSHINE'09*, ser. LNCS, vol. 22. Springer, 2009, pp. 280–296.

[13] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[14] D. Khan, N. Navet, B. Bavoux, and J. Migge, "Aperiodic traffic in response time analyses with adjustable safety level," in *Proceedings of ETFA'09*. IEEE Computer Society, 2009, pp. 1–9.

[15] K. Kim, J. Díaz, L. Bello, J. López, C.-G. Lee, and S. L. Min, "An exact stochastic analysis of priority-driven periodic real-time systems and its approximations," *IEEE Trans. Computers*, vol. 54, no. 11, pp. 1460 – 1466, 2005.

[16] J.-Y. Le Boudec and P. Thiran, *Network calculus: A theory of deterministic queuing systems for the Internet*. Springer, 2001, vol. 2050 of LNCS.

[17] J. P. Lehoczky, "Real-time queueing network theory," in *Proceedings of RTSS'97*. IEEE Computer Society, 1997, pp. 58 –67.

[18] ——, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proceedings of RTSS'90*. IEEE Computer Society, 1990, pp. 201–213.

[19] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[20] J. M. López, J. L. Díaz, J. Entrialgo, and D. García, "Stochastic analysis of real-time systems under preemptive priority-driven scheduling," *Real-Time Systems*, vol. 40, no. 2, pp. 180–207, 2008.

[21] S. Manolache, P. Eles, and Z. Peng, *Real-Time Applications with Stochastic Task Execution Times: Analysis and Optimisation*. Secaucus, NJ, USA: Springer, 2007.

[22] D. Maxim, L. Santinelli, and L. Cucu-Grosjean, "Improved sampling for statistical timing analysis of real-time systems," in *the 4th Junior Researcher Workshop on Real-Time Computing*, Toulouse, France, 2010.

[23] S. Quinton, M. Hanke, and R. Ernst, "Formal analysis of sporadic overload in real-time systems," 2012, to appear in the Proceedings of DATE'12.

[24] B. Raman, G. Quintin, W. T. Ooi, D. Gangadharan, J. Milan, and S. Chakraborty, "On buffering with stochastic guarantees in resource-constrained media players," in *Proceedings of CODES+ISSS'11*. ACM, 2011, pp. 169–178.

[25] K. S. Refaat and P.-E. Hladik, "Efficient stochastic analysis of real-time systems via random sampling," in *Proceedings of ECRTS'10*. IEEE Computer Society, 2010, pp. 175–183.

[26] L. Santinelli, P. Meumeu Yomsi, D. Maxim, and L. Cucu-Grosjean, "A component-based framework for modeling and analyzing probabilistic real-time systems," in *Proceedings of ETFA'11*. IEEE Computer Society, 2011, pp. 1 –8.

[27] T.-S. Tia, Z. Deng, M. Shankar, M. F. Storch, J. S. 0002, L.-C. Wu, and J. W.-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Proceedings of RTAS'95*. IEEE Computer Society, 1995, pp. 164–173.

[28] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. B. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem - overview of methods and survey of tools," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, 2008.

[29] J. Xie and Y. Jiang, "Stochastic network calculus models under max-plus algebra," in *Proceedings of GLOBECOM'09*. IEEE Computer Society, 2009, pp. 1–6.