A tool for optimal design of soft–real time systems

Luigi Palopoli, Luca Abeni, Daniele Fontanelli
Dipartimento di Scienze e Ingegneria dell’Informazione
University of Trento, Trento, Italy
\{luigi.palopoli,luca.abeni,daniele.fontanelli\}@disi.unitn.it

Abstract—In recent years a series of important achievements have paved the way for the introduction of probabilistic analysis in the area of soft–real–time systems design. In this paper, we present an extensible design tool – called PROSIT – that facilitates the access to this technology for a potentially large number of researchers and industrial practitioners. The tool enables probabilistic analysis of the temporal performance of a real–time task under fixed priority and resource reservations scheduling. For resource reservations, the tool also offers an automatic procedure for the synthesis of scheduling parameters that optimise a quality metric related to the probabilistic behaviour of the tasks.

Keywords-Soft real–time systems, Probabilistic Guarantees

I. INTRODUCTION

In the classic context of hard real–time systems, a system consists of a set of tasks; each one generates a stream of jobs that have to terminate before a deadline. For soft real–time systems this requirement is too strict and can be revisited in a probabilistic sense. One of the possibilities is to specify a probability for the termination of a task within a deadline, or more generally, a desired distribution of the delays.

It is argued that the Quality of different type of industrially relevant applications can be easily related to such probabilistic metrics. This consideration, along with the ever-increasing variability of modern real–time applications, has stimulated an intense research on probabilistic real–time systems. A very important line of work is on the probabilistic generalisation of analysis technique initially developed in the hard real–time domain such as time demand analysis [1] and computation of the response time of tasks scheduled with fixed priority and with stochastic computation time [2], [3], the inter-arrival time [4] or both [5].

Other papers assume reservation based scheduling [6]. In a reservation-based scheduler tasks enjoy the temporal isolation property, which allows the designer to decouple the behaviour of the different tasks. Based on this, Abeni et al. [7] have proposed an analysis technique based on on queueing theory for reservation-based schedulers. Recently, approximated solution techniques have been proposed to decrease the complexity of the algorithm used to compute the probability of meeting the deadline [8], [9], while Mills and Anderson [10] have looked at the problem from the perspective of multi-processor systems.

Paper Contribution. The time is ripe to make some of the most effective techniques for probabilistic analysis of real–time systems available to a wide community of researchers and industrial practitioners. This is the purpose of the PROSIT tool presented in this paper. The tool design rests on two pillars. The first one is the analysis of the stochastic behaviour of real–time tasks. In the recent literature, it is shown that such analysis can be reduced to that a of particular class of discrete–time Markov Chains (DTMC) called Quasi-Birth-Death process [11]; this holds both for fixed priority scheduling [2], [3] and for resource reservations [8]. This property allowed us to use some of the most efficient numeric and analytic algorithms [12] as a basis for the implementation of the system analysis. The second pillar is the definition of the Quality as a function of the probability of meeting deadlines (or more generally of the distribution of the delays). This allowed us to revisit in a probabilistic framework the ideas proposed in such frameworks as QRAM [13], and to develop algorithms for the optimal choice of the scheduling parameters. The solution of the optimisation problem is particularly efficient when we choose as a global quality metric the worst case Quality between those experienced by the applications.

II. PROBLEM PRESENTATION

Application Model. In this paper, we consider a set of independent applications each one composed of various real–time tasks. The applications share the same computing resources, but do not interact in other ways. In this paper, we restrict our focus to single-task applications, reserving extensions to multi-task applications for our future work.

A real-time task $\tau_i$ is a stream of jobs $J_{i,k}$; job $J_{i,k}$ arrives (becoming executable) at time $r_{i,k}$, finishes at time $f_{i,k}$ after executing for a time $c_{i,k}$, and is characterised by a deadline $d_{i,k} = r_{i,k} + D_i$, (where $D_i$ is the relative deadline of $\tau_i$). We consider a generalisation of the traditional notion of deadline, called probabilistic deadline [7]. A probabilistic deadline is a pair $\langle \delta_i, p_i \rangle$, where $p_i$ is the steady-state probability of respecting a relative deadline $\delta_i$. 

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n ICT-2011-288917 DALi - Devices for Assisted Living and by the HYCON2 NoE, under grant agreement FP7-ICT-257462.
\( (d_{i,k} = r_{i,k} + \delta_i) \). The probabilistic deadline is respected if \( \Pr \{ f_{i,k} \leq r_{i,k} + \delta_i \} \geq \mu_i \). It is possible to specify a sequence of probabilistic deadlines for \( \tau_i \), \( \{ \delta_1^{(i)}, p_1^{(i)} \}, \ldots, \{ \delta_{H_i}^{(i)}, p_{H_i}^{(i)} \} \), where \( p_{H_i}^{(i)} \geq p_{j}^{(i)} \) if \( \delta_{H_i}^{(i)} > \delta_{j}^{(i)} \), formulating in this way a specification on the distribution of the delays.

The computation time of each job \( c_{i,k} \) is a stochastic process \( U_i \) described by the probability mass function (PMF) \( U_i(c) = \Pr \{ c_{i,k} = c \} \). Likewise, the inter-arrival time (the distance between two subsequent activations) is assumed to be a stochastic process \( I_i \) with PMF \( I_i(t) = \Pr \{ r_{i,k+1} - r_{i,k} = t \} \). As a special case, when the \( I_i(t) \) distribution is given by a Kronecker delta centred in \( T_i \) \( (I_i(t) = 1(t - T_i)) \) the task is periodic of period \( T_i \).

**Application Quality.** While probabilistic deadlines effectively capture the real-time constraints, the quality perceived by the user of the application is not immediately expressed by such quantities. Much more useful is a quality index \( \mu_i \), which is necessarily application specific and is related to the design choices. This index is related to the probabilistic deadlines by a functional dependence. In the simplest case, we can consider a single probabilistic deadline where \( \delta_i \) is equal to the relative deadline \( D_i \). In this case the index \( \mu_i \) is a monotone non decreasing function of probability \( p_i \). More generally, if we consider a sequence of probabilistic deadlines, \( \mu_i \) is a function of the vector \( \{ p_1^{(i)}, \ldots, p_{H_i}^{(i)} \} \).

Our notion of a Quality function is inspired to the QRAM framework [13]. However, QRAM assumes a functional dependence between scheduling parameters and \( \mu_i \), which can be difficult to identify. On the contrary, we relate the quality to the probabilistic temporal behaviour of the task, which depends on the scheduling parameters in non-obvious ways. This allows the designer to reason about the system quality in a natural conceptual framework without committing to any particular scheduling policy.

As an example, in the domain of control applications, a quality index of this kind could be the steady state covariance of the controlled plant state, which is known to be a function of the distribution of the delays [14]. Likewise, for a media processing application, we can define \( \mu_i \) as the PSNR or SSIM between the original media and the one reproduced by the application [15]. Under the assumption that video frames decoded by jobs finishing late are not displayed, when a deadline is missed the PSNR or SSIM is computed using the last frame decoded in time [16] and degrades with the frequency of this event.

**Scheduling.** Although fixed priority scheduling [17] is probably not the best possible choice for soft real-time systems, its enormous popularity suggested us to consider it within the PROSIT framework.

A more reputable scheduling choice for this domain are the so called CPU Reservations [6]: each task \( \tau_i \) is associated with a reservation \( (Q_i^t, T_i^r) \), meaning that \( \tau_i \) is guaranteed to execute for \( Q_i^t \) (budget) time units in every interval of length \( T_i^r \) (reservation period). The PROSIT tool operates with any reservation scheduler as long as it respects the temporal isolation property: the amount of computation time \( Q_i^t \) is reserved to \( \tau_i \) regardless of the behaviour of the other tasks. The ratio \( B_i = Q_i^t / T_i^r \) is termed bandwidth and corresponds to the fraction of CPU allocated to the task.

**The analysis and the synthesis problem.** The first problem we address, called the Analysis Problem, can be shortly described as follows: given a set of applications, each one characterised by its probability distributions, a quality function, a scheduling algorithm, a set of scheduling parameters \( \text{par}_i \) and a sequence of probabilistic deadlines, decide if the steady state probabilities \( \{ p_1^{(i)}, \ldots, p_{H_i}^{(i)} \} \) are respected and compute the Quality \( \mu_i \) for each of the applications. The scheduling parameters \( \text{par}_i \) are given by the priorities \( \text{prior}_i \), for a fixed priority scheduler and by the pair \( (Q_i^t, T_i^r) \) for a resource reservation scheduler.

In the analysis problem the scheduling parameters are assumed chosen by the designer. The synthesis problem is different since it requires the computation of optimal scheduling parameters. The idea is that a choice of scheduling parameters determines a set of steady state distribution and hence a different quality for each of the applications. The different qualities can be collected to form a global quality function \( f(\mu_1, \ldots, \mu_n) \). The synthesis problem can be expressed as \( \max_{\text{par}_1, \ldots, \text{par}_n} f(\mu_1, \ldots, \mu_n) \) subject to \( \mu_i \geq I_i \) and to a schedulability condition. An example of schedulability condition for resource reservations is CPU time allocated to the different tasks cannot exceed a specified value \( \sum_{i=1}^n B_i \leq U^{\text{lab}} \) [7].

**III. THE PROSIT TOOL: AN OVERVIEW.**

The typical workflow in using the PROSIT tool are sketched in Figure 1, which represent two use cases related to the solution of the problems described above.

**The analysis problem.** For the analysis problem (top half of the figure), the designer is required to provide the definition of the task set that he/she wishes to analyse.

If the chosen scheduling algorithm is resource reservation, in order to analyse one task the user only needs to provide its timing requirements (distribution of the interarrival time \( I_i(t) \) and of its computation time \( U_i(c) \)) and its scheduling parameters \( (Q_i^t, T_i^r) \). This is because the temporal isolation property allows decoupling the analysis of the different tasks. On the contrary, if fixed priority scheduling is used, the analysis is possible only for periodic task and requires the knowledge of the temporal requirements of the task and of all those having a higher priority [2], [3]. The distributions can either be chosen from a library (e.g., uniform, beta, Gaussian, etc.) customising the required parameters or specified in a file. In addition to specifying the task set, the user also chooses a solution algorithm for the probabilistic analysis and the Quality evaluator (i.e., the
function $\mu_i$ relating the application quality to the probabilistic deadlines). This information can be inserted using command line parameters or a system specification file.

As a result of the tool invocation, a C++ object is generated that captures all the information required for the analysis. One of the methods of this object (solve) computes the probabilistic deadlines. This is a virtual method, which is overloaded with the specific solution strategy chosen by the user. The quality corresponding to the distribution of the probabilities for the probabilistic deadlines is computed by a different method (Quality), which is itself virtual to allow for different quality models. The tool prints the result and the computation time to screen (and/or to a file).

**The synthesis problem.** The solution of the synthesis problem is currently available only for the resource reservations. The tool requires a system specification file containing a description of all the temporal information of all the tasks (see Figure 1). For each task, the user specifies the temporal parameters, the server period $T^s_i$, the solution algorithm, the Quality function and the minimum required value for the quality. The budget $Q^*_i$ is computed by the optimisation algorithm. One could legitimately argue over the choice of fixing $T^s_i$ and leaving $Q^*_i$ as decision variable. The motivation is rooted in the philosophy of the resource reservations, in which $T^s_i$ is used to control the granularity of resource allocation, while $Q^*_i$ is used to control the bandwidth.

In the system specification file, the user also specifies the global quality function $f$. This function composes the quality associated with each application, which is a non-negative real number. Possible choices are the infinity norm $f(\mu_1, \ldots, \mu_n) = \max_{i=1}^n \mu_i$ - and the one norm $f(\mu_1, \ldots, \mu_n) = \sum_{i=1}^n \mu_i$. After the tool execution is started, the parser generates C++ object instances for each task and hands them over to the optimiser (which is itself a C++ object). The optimisation algorithm iteratively calls the quality method of the task (and indirectly the solve method) to compute the quality associated to a choice of decision variables or to estimate the gradient and eventually produces the optimal choice of parameters.

**Definition of the Problem.** A possible workflow to derive the information required by the PROSIT tool for its operations is shown in Figure 2. The system specification file is an XML file and can be inserted by any XML editor (or by a GUI in a future development). An excerpt of an example file is shown in Figure 3. The first line defines the type of problem (in the example a synthesis problem based on the infinity norm maximisation). Then follows a definition of the different tasks, each one associated with a symbolic name and a type. The latter can be chosen from a library of existing types. A task type is associated with a solution strategy (in the example the analytic bound [9]) and with the quality computation (in the example is user-provided). In the definition of the task we specify interarrival time (fixed in the example because the task is periodic) and computation time (in the example described by a Probability Mass Function contained in an external file). In addition, we specify the scheduling algorithm (reservation) and the user defined scheduling parameters (the reservation period). The user-provided Quality model is specified by a type (chosen in a library) and by a set of parameters. In the example, we adopt a simple linear model (see Section VI for more details). In the same entry, we also define a lower bound for the quality of service. An important problem is how to derive the computation requirements of the task (distribution of computation time, period, distribution of the interarrival time). This is relatively easy if the designer is in control of the source code and can instrument it with probes pinpointing the start and the termination of each job.

The task is much harder for legacy applications and can be accomplished using two external modules: a tracer that operates inside the kernel and notes all the events related to the application and an event analyser, which detects periods and estimates the distribution of the computation time.

**API.** The tool is based on a C++ library designed to be flexible and extensible. Most of the features of the tool are exposed to the software developer through an API for the possible benefits of using some of the library components outside the PROSIT tool. For instance, one could use the library for an admission test based on probabilistic deadlines.

The library can be easily extended in several directions by: 1) the definition of new solution algorithms for probability computation, 2) the definition of new quality metrics, 3) the definition of different distribution types, 4) the definition of different optimisation algorithms. In some cases, such extensions are made by sub-classing and redefinition of a few methods. In others (e.g., for the quality metrics), the user has to define her/his own function objects using a factory design pattern [18], along with a couple of auxiliary functions for XML parsing, which make the newly defined extensions readily accessible within the tool.

**IV. ALGORITHMS ANALYSIS AND SYNTHESIS**

**Probabilistic Analysis.** The cornerstone of our tool is the probabilistic analysis. For the sake of brevity, let us focus
on periodic task using resource reservation scheduling. As shown in the literature [7], a stochastic process $v_{i,j}$ can be introduced to model the amount of time to be executed after the arrival of the $j^{th}$ job $J_{i,j}$. The initial backlog $v_{i,0}$ is obviously equal to $c_{i,0}$ and $v_{i,j+1}$ can be easily computed as

$$v_{i,j+1} = \max\{0, v_{i,j} - N_i Q_i^s\} + c_{i,j+1}$$  \hspace{1cm} (1)

where $N_i = T_i / T_i^s$ is an integer number describing the number of reservation periods contained in the task period $T_i$. This model is easily generalised to the case of aperiodic tasks (making the conservative simplification that the activation of each job is considered only at the beginning of a reservation period) [8] and describes the evolution of a discrete-time Markov Chain (DTMC). A very important feature of this model is its recursive structure, leading to which qualifies the DTMC as a Quasi-Birth-Death (QBD) process. As discussed by Díaz et al. [2], if one considers the probabilistic a set of tasks scheduled by fixed priority, the analysis reduces to one of a DTMC having the same structure. This consideration are of the greatest importance because very effective numeric solutions exists for QBD processes. In particular, in the PROSIT tool we have implemented logarithmic reduction [19] and cyclic reduction [12]. The two implementations can serve as a template for the implementation for any of the plethora of algorithms documented in the literature [11].

Given the specific structure of the QBD, it is also possible to compute an analytical bound for the probability of deadline miss. The algorithm is shown in [9]; as discussed in the paper and shown in the experimental section this bound is particularly useful in the synthesis problem for the computation of suboptimal solutions of acceptable quality with substantial savings of time.

**Synthesis.** As discussed in Section II, the synthesis amounts to the solution of optimisation algorithm, which in the current version of the tool is possible only for resource reservations. The solution is very efficient when the infinity norm is used and the following assumption can be made: the quality increases if the budget reserved to the application, and hence the probability of meeting the deadline increases. In all cases of practical relevance that we have examined, this assumption is easily verified.

The monotonicity of the function allows us to apply the efficient solution algorithm reported in Procedure 1. The first lines of the algorithm (3 through 5) are to verify if the total bandwidth required to attain the lower bounds of the specification exceeds 100%, the problem being unfeasible in this case. The search for the optimal solution is reduced to within two bounds (line 6): the lower one ($\mu$) derives form the lower bound constraints, while the higher one is obtained
by assigning 100% of the bandwidth to the task \( Q^\mu = T^\mu \).
The method invQuality computes the budget required to attain a specified level of quality: since the Quality is assumed monotone increasing, the inversion can be carried out by a simple dichotomic search. The operation can be expensive because it entails repeated calls to the solve method. The code segment between line 7 and 15 computes the budget required to each task for the upper bound \( \pi \). If some of the tasks is constrained to a lower bound \( \mu \), higher than \( \pi \), this task is removed from the subsequent search phases and it is allocated a bandwidth sufficient to attain its lower bound. The execution of the algorithm is terminated if the algorithm converges to the optimum.

\[ \text{By \( \beta_{a,b} \) we denote the beta distribution} \]
\[ \beta_{a,b}(c) = \frac{1}{B(a,b)} c^{a-1}(1-c)^{b-1} \text{ defined over the range} \]
\[ \text{where} \ r(c) = \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \text{.}\]

\[ \text{The Quality is, in this example, a very simple function of the probability} \]
\[ p \text{ of meeting a deadline set equal to the period. Specifically,} \]
\[ \text{linear}_{p_{\text{min}}, p_{\text{max}}, \alpha}(p) \text{ denotes a function equal to} \]
\[ 0 \text{ if} \ p_{\text{min}} < p < p_{\text{max}} \text{ and} \]
\[ p_{\text{min}} \text{ for} \ p < p_{\text{min}} \text{ and} \]
\[ 1 \text{ for} \ p > p_{\text{max}} \text{.}\]

\[ v \text{ for CR and} \]
\[ v_{\text{max}} \text{ for} \]
\[ v_{\text{min}} \text{.}\]

\[ \frac{1}{B(a,b)} c^{a-1}(1-c)^{b-1} \text{ defined over the range} \]
\[ c_{\text{min}} \text{ and} \ c_{\text{max}} \text{ where} \]
\[ r(c) = \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \text{.}\]

\[ \text{The Quality is, in this example, a very simple function of the probability} \]
\[ p \text{ of meeting a deadline set equal to the period. Specifically,} \]
\[ \text{linear}_{p_{\text{min}}, p_{\text{max}}, \alpha}(p) \text{ denotes a function equal to} \]
\[ 0 \text{ if} \ p_{\text{min}} < p < p_{\text{max}} \text{ and} \]
\[ p_{\text{min}} \text{ for} \ p < p_{\text{min}} \text{ and} \]
\[ 1 \text{ for} \ p > p_{\text{max}} \text{.}\]

\[ v \text{ for CR and} \]
\[ v_{\text{max}} \text{ for} \]
\[ v_{\text{min}} \text{.}\]

\[ \frac{1}{B(a,b)} c^{a-1}(1-c)^{b-1} \text{ defined over the range} \]
\[ c_{\text{min}} \text{ and} \ c_{\text{max}} \text{ where} \]
\[ r(c) = \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \text{.}\]

\[ \text{The Quality is, in this example, a very simple function of the probability} \]
\[ p \text{ of meeting a deadline set equal to the period. Specifically,} \]
\[ \text{linear}_{p_{\text{min}}, p_{\text{max}}, \alpha}(p) \text{ denotes a function equal to} \]
\[ 0 \text{ if} \ p_{\text{min}} < p < p_{\text{max}} \text{ and} \]
\[ p_{\text{min}} \text{ for} \ p < p_{\text{min}} \text{ and} \]
\[ 1 \text{ for} \ p > p_{\text{max}} \text{.}\]

\[ v \text{ for CR and} \]
\[ v_{\text{max}} \text{ for} \]
\[ v_{\text{min}} \text{.}\]

\[ \frac{1}{B(a,b)} c^{a-1}(1-c)^{b-1} \text{ defined over the range} \]
\[ c_{\text{min}} \text{ and} \ c_{\text{max}} \text{ where} \]
\[ r(c) = \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \text{.}\]

\[ \text{The Quality is, in this example, a very simple function of the probability} \]
\[ p \text{ of meeting a deadline set equal to the period. Specifically,} \]
\[ \text{linear}_{p_{\text{min}}, p_{\text{max}}, \alpha}(p) \text{ denotes a function equal to} \]
\[ 0 \text{ if} \ p_{\text{min}} < p < p_{\text{max}} \text{ and} \]
\[ p_{\text{min}} \text{ for} \ p < p_{\text{min}} \text{ and} \]
\[ 1 \text{ for} \ p > p_{\text{max}} \text{.}\]

\[ v \text{ for CR and} \]
\[ v_{\text{max}} \text{ for} \]
\[ v_{\text{min}} \text{.}\]

\[ \frac{1}{B(a,b)} c^{a-1}(1-c)^{b-1} \text{ defined over the range} \]
\[ c_{\text{min}} \text{ and} \ c_{\text{max}} \text{ where} \]
\[ r(c) = \frac{c - c_{\text{min}}}{c_{\text{max}} - c_{\text{min}}} \text{.}\]

\[ \text{The Quality is, in this example, a very simple function of the probability} \]
\[ p \text{ of meeting a deadline set equal to the period. Specifically,} \]
\[ \text{linear}_{p_{\text{min}}, p_{\text{max}}, \alpha}(p) \text{ denotes a function equal to} \]
\[ 0 \text{ if} \ p_{\text{min}} < p < p_{\text{max}} \text{ and} \]
\[ p_{\text{min}} \text{ for} \ p < p_{\text{min}} \text{ and} \]
\[ 1 \text{ for} \ p > p_{\text{max}} \text{.}\]

\[ v \text{ for CR and} \]
\[ v_{\text{max}} \text{ for} \]
\[ v_{\text{min}} \text{.}\]
while the exact approaches are preferable when an offline execution of the tool allows for a more precise solution.

VII. CONCLUSIONS

We have described a software tool for probabilistic design of real–time systems called PROSIT, shown concrete use cases and discussed its algorithmic foundations. An intense development activity covering a significant number of features of the tool, such as a full support for fixed priority, and different optimisation algorithms, is currently ongoing. From the modelling point of view, we are looking at different types of real–time applications (e.g., multi-task and distributed applications); the analysis of real applications will extend the library of available quality functions.

REFERENCES


