Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (ASH 2015), June 15-18, 2015, Montreal, QC, Canada.

Adaptive Mechanisms for Component-Based Real-Time Systems

Giorgio Buttazzo[1] and Luca Santinelli[2]
[1] *Scuola Superiore Sant'Anna, Pisa, Italy*, g.buttazzo@sssup.it
[2] *ONERA The French Aerospace Lab, Toulouse, France*, luca.santinelli@onera.fr

***Abstract*—When a common computing platform is shared by several software activities (tasks), the interference generated by the concurrent access to computational resources introduces unpredictable delays on task execution that may jeopardize the correct behavior of the controlled system. In safety-critical systems, an effective method for limiting such an interference is resource partitioning (or resource reservation), according to which each task is assigned a fraction of the shared resource (bandwidth) and executes in isolation as it were executing alone on a system with less resources.**

The advantage of this approach is that the response time of each task does not depend on the execution behavior of the other activities, but only on its own computational demand and on the amount of allocated resource. However, the resulting system performance strongly depends on a correct resource allocation, that is the size of the partitions.

Given the dynamic behavior of certain applications and the difficulty of predicting their resource needs, adaptive resource management is crucial for changing the allocation to the actual resource requirements when they are not correctly estimated.

This paper presents an adaptive resource reservation algorithm for partitioning the processor among concurrent real-time tasks and illustrates the analysis for computing the probability of meeting the timing constraints specified on the application tasks, and evaluating the changes on system partitions.

Keywords*—*Adaptive systems, Component-based systems, Real-time software, Resource reservation, Resource partitioning, Probabilistic analysis.

I. Introduction

Complex embedded software for aerospace applications consists of many concurrent activities (tasks) that execute on a computing platform and interact through shared resources (e.g., processor, memory, and I/O devices). Due to the complex structure of control applications, the dependency of the source code on input data and the interaction with the other tasks, the execution time of each task is subject to high variations, which are very difficult to predict off line. In addition, other architecture features of the computing platform (such as cache memory, pipeline and pre-fetch mechanisms, DMA and interrupts), concur in increasing such a variability, making the worst-case execution times much higher than the average-case ones. Also, operating system mechanisms such as scheduling, synchronization, and communication, may introduce additional delays in task executions.

As a consequence, the execution time of a task can be described by a distribution function, which ranges between a *Best-Case Execution Time* (BCET) to a *Worst-Case Execution Time* (WCET). Figure 1 shows an example of execution time distribution. Due to the reasons described above, the WCET of a task is much higher than its BCET, making the analysis either very pessimistic or unsafe.

Fig. 1. Execution time density and normalize frequency, with the best-case and the worst-case execution times.

An effective method for reducing task interference and make task execution times more predictable is to partition a resource into fractions and allocate each fraction to a task. For example, a simple mechanism for partitioning the processor computation time among different tasks is to divide the time into slots of fixed length and allocate each task to a different slot, in such a way that all the timing constraints specified by the application are satisfied. According to this method, the schedule is constructed off-line and cyclically executed by a runtime executive. The major problem of this approach is a lack of flexibility for online changes. In fact, once slots are defined and assigned to tasks, even small variations in the schedule (as the insertion of new tasks, or changes in their activation rates) may require a complete task reallocation.

A more flexible partitioning of the processor time can be implemented by a resource reservation mechanism, according to which each task (or task group) is executed within a periodic server S_i that executes up to Q_i units of time (*server budget*) every period P_i (*server period*). If the served task is longer than Q_i units, the remaining part is postponed in the next server periods. In this way, a reservation server S_i guarantees that a fraction $\alpha_i = Q_i/P_i$ (*server bandwidth*) is allocated to a given task (or task group).

Note that a task receiving a fraction α_i of the total processor bandwidth behaves as it was executing alone on a slower processor (virtual processor) with a speed equal to α_i times the full speed. Resource reservation can be used to implement a hierarchical scheduling scheme, where multiple applications can be handled within different reservation servers, each using a local scheduler, while all the servers are handled by a global scheduler on the physical processor.

A resource reservation technique for fixed priority systems was first presented by Mercer, Savage and Tokuda [1]. Fixed priority scheduling, however, has the disadvantage that an arbitrary real-time application cannot exploit the full processing bandwidth. Liu and Layland [2] have shown that the best fixed-priority assignment to achieve the feasibility of a set of periodic tasks with deadlines equal to periods is the Rate Monotonic (RM) algorithm, which assigns priorities proportionally to task activation rates. Even using the RM algorithm, a set of arbitrary real-time periodic tasks can be feasibly scheduled on a processor only if their total utilization does not exceed 69 percent of the total processor bandwidth.

Using the Earliest Deadline First (EDF) scheduling algorithm, which schedule tasks based on their absolute deadlines, Liu and Layland [2] proved that a set of arbitrary real-time periodic tasks can exploit the full processor bandwidth. Under EDF scheduling,

resource reservation can be efficiently implemented through the Constant Bandwidth Server (CBS) [3], [4], which recently has also introduced in the Linux mainline kernel [5].

The main advantage of resource reservation is that the response time of each task does not depend on the execution behavior of the other activities, but only on its own computational demand and on the amount of allocated bandwidth. On the other hand, the resulting system performance strongly depends on a correct bandwidth allocation, which is very difficult to achieve, given the dynamic behavior of complex applications and the difficulty of predicting their resource needs. Therefore, implementing an adaptive reservation mechanism is crucial for tuning the allocated bandwidth based on the actual resource requirements.

When the available computational resources are not sufficient to guarantee the application in the worst-case scenario, smarter techniques are needed to sense the current state of the system and react as a consequence to keep the system performance at a desired level or, when this is not possible, degrade it in a controlled fashion. With timing constraints and available resource trade-offs, the system is able to adapt to changes.

Paper contribution. This paper presents an adaptive resource reservation algorithm for partitioning the processor among concurrent real-time tasks and illustrates the analysis for computing the probability of meeting the timing constraints specified on the application tasks. The paper presents also probabilistic schedulability analyses to guarantee the adaptive behavior of the systems. With the adaptive resource reservation algorithm and the probabilistic schedulability analyses we guarantee timing properties of adaptive safety-critical systems.

Paper organization. The remainder of this paper is organized as follows. Section II presents the computational model used to describe real-time applications and introduces the key definitions that are used in the paper. Section III summarizes the theoretical background needed to understand the schedulability analysis proposed in the paper. Section IV presents the adaptive framework for coping with load variations and wrong resource assignments. Section V describes the probabilistic analysis for guaranteeing application deadlines. Finally Section VI states our conclusions and future work.

II. System model

In this paper, we consider a generic application consisting of n real-time tasks, denoted as $\tau_1, \ldots, \tau_n$. Each task τ_i is characterized by an *expected computation time* C_i, a *relative deadline* D_i, and a *minimum inter-arrival time* T_i. Each task τ_i can be activated multiple times on different data, thus generating a sequence of instances (or *jobs*) $\tau_{i,1}, \tau_{i,2}, \ldots$. Considering that every job of a task runs on different input data, the computation time of a task τ_i can be described by a distribution of execution times, spanning from a *best-case execution time* to a *worst-case execution time*. Similarly, since task activation is triggered by events in the environment, the inter-arrival time of consecutive jobs of a task τ_i is also described by distribution of separation intervals, whose minimum value is represented by T_i. The ratio $\overline{U}_i = C_i/T_i$ is referred to as the *task utilization factor*, whereas the sum

$$\overline{U} = \sum_{i=1}^{n} \overline{U}_i \tag{1}$$

is called the *processor utilization factor* or total task set utilization. Each individual job $\tau_{i,k}$ is characterized by a release time $r_{i,k}$, an actual execution time $e_{i,k}$, and an absolute deadline $d_{i,k} = r_{i,k} + D_i$, which is the maximum time the job is allowed to complete in order to guarantee a given performance. The absolute finishing time at which a job $\tau_{i,k}$ completes its execution is denoted as $f_{i,k}$, whereas the job response time is defined as $R_{i,k} = f_{i,k} - r_{i,k}$. The response time of task τ_i is defined as the maximum response time among all its jobs, that is $R_i = \max_k \{R_{i,k}\}$.

A task is said to be *schedulable* by a specific scheduling algorithm if all its jobs are guaranteed to complete by their deadlines, for any activation sequence respecting the minimum inter-arrival time. A task is said to experience an *execution overrun* if the actual execution time of a job exceeds its expected computation time C_i.

To avoid that the execution overruns experienced by a task affect the schedulability of other tasks, each task τ_i is executed within a dedicated reservation server S_i with budget Q_i and period P_i. The ratio $\alpha_i = Q_i/P_i$ is referred to as the *server bandwidth*. We assume that all the servers are scheduled by the EDF algorithm.

A. Probabilistic Task Model

The probabilities allows to better characterize system variability and to better cope with the different safety levels that today's safety-critical systems exibits. In a probabilistic real-time framework, task WCET and task *Minimum Inter-arrival Time (MIT)* can be represented as discrete distributions[1] with the multiple values they can assume together with the probability of happening for those values. For task τ_i the *probabilistic WCET* (pWCET) distribution is $\mathcal{C}_i$, its probability distribution functions (pdf) $f_{\mathcal{C}_i}$ is

$$f_{\mathcal{C}_i} = \begin{pmatrix} C_i^0 = C_i^{min} & C_i^1 & \cdots & C_i^{k_i} = C_i^{max} \\ f_{\mathcal{C}_i}(C_i^{min}) & f_{\mathcal{C}_i}(C_i^1) & \cdots & f_{\mathcal{C}_i}(C_i^{max}) \end{pmatrix}; \tag{2}$$

and τ_i *probabilistic MIT* (pMIT) distribution is $\mathcal{T}_i$, which in terms of pdf $f_{\mathcal{T}_i}$ looks like

$$f_{\mathcal{T}_i} = \begin{pmatrix} T_i^0 = T_i^{max} & T_i^1 & \cdots & T_i^{l_i} = T_i^{min} \\ f_{\mathcal{T}_i}(T_i^{max}) & f_{\mathcal{T}_i}(T_i^1) & \cdots & f_{\mathcal{T}_i}(T_i^{min}) \end{pmatrix}, \tag{3}$$

where $\sum_{j=0}^{k_i} f_{\mathcal{C}_i}(C_i^j) = 1$ and $\sum_{j=0}^{l_i} f_{\mathcal{T}_i}(T_i^j) = 1$. Here, (k_i+1) and (l_i+1) denote the number of pWCET and pMIT representing task τ_i, respectively. The values of pMIT are ordered in an opposite manner than those of pWCET, for the sake of readability and the ease representation of next mathematical expressions. In the following, calligraphic uppercase letters are used to refer to probabilistic distributions, while non calligraphic letters are used for single value parameters.

Since distributions $\mathcal{C}_i$ and $\mathcal{T}_i$ include worst-case values, they are a safe representation of the task behavior. Furthermore, by applying worst-case distributions we also guarantee statistical independence between tasks and task instances. Indeed, any possible dependency is already included into both $\mathcal{C}_i$ and $\mathcal{T}_i$, hence it cannot affect those distributions anymore[2].

All the parameters C_i^k and T_i^l are given with the interpretation that the execution requirement of each job $\tau_{i,j}$ is described by $\mathcal{C}_i$ (worst-case execution requirement), where for each value C_i^k, $f_{\mathcal{C}_i}(C_i^k)$ is its probability of occurrence. Function $f_{\mathcal{T}_i}(T_i^k)$ gives the probability that the arrival time of the next job of task τ_i is equal to T_i^k.

[1] Since the execution time or the inter-arrival time of a program can only take discrete values that are multiples of the processor clock cycle.

[2] In case of independence the joint probability is $\mathbb{P}\{\mathcal{C}_i = x_i, \mathcal{C}_j = x_j\} = \mathbb{P}\{\mathcal{C}_i = x_i\} \cdot \mathbb{P}\{\mathcal{C}_j = x_j\}$, since for the conditional probability it is $\mathbb{P}\{\mathcal{C}_i = x_i | \mathcal{C}_j = x_j\} = \mathbb{P}\{\mathcal{C}_i = x_i\}$.

Alternatively, the worst-case execution time distribution, as well as the minimum inter-arrival distribution, can be specified using its Cumulative Distribution Function (CDF), denoted by $F_{\mathcal{C}_i}(x)$ or $F_{\mathcal{T}_i}(x)$, where $F_{\mathcal{C}_i}(x) = \sum_{c=0}^{x} f_{\mathcal{C}_i}(c) \equiv \mathbb{P}\{\mathcal{C}_i \leq x\}$. The inverse of the CDF, that is 1-CDF, is denoted as $F'_{\mathcal{C}_i}(x) = 1 - \sum_{c=0}^{x} f_{\mathcal{C}_i}(c) \equiv \mathbb{P}\{\mathcal{C}_i \geq x\}$; equivalently $F'_{\mathcal{T}_i}(x) = 1 - \sum_{c=0}^{x} f_{\mathcal{T}_i}(c) \equiv \mathbb{P}\{\mathcal{T}_i \geq x\}$. Function 1-CDF exploits the probabilities in terms of exceeding thresholds, that is, $F'(x)$ is the probability of exceeding a defined threshold x.

The *probabilistic task utilization* becomes a distribution

$$\overline{U}_i = \frac{\mathcal{C}_i}{\mathcal{T}_i}, \tag{4}$$

whose values are the ratios between $\mathcal{C}_i$ and $\mathcal{T}_i$ values, and the probabilities are the product of $\mathcal{C}_i$ and $\mathcal{T}_i$ probabilities. The *probabilistic application utilization* is

$$\overline{U} = \otimes_{\tau_i \in \Gamma} \overline{U}_i \tag{5}$$

composes all the task utilization with the convolution operator, due to the independence among tasks.

III. Theoretical background

In a system consisting of n reservations, each implemented by a periodic server with bandwidth U_i, the feasibility of the schedule can be guaranteed as shown in Section V. In particular, if reservations are scheduled by RM, the schedulability of the system can be verified by Equation (9); if they are scheduled by EDF, schedulability can be verified by Equation (11).

The feasibility analysis of a real-time task within a reservation is more complex and requires the precise knowledge of how time is made available by the server. Although a reservation is typically implemented using a periodic server characterized by a budget Q_s and a period P_s, there are cases in which temporal isolation can be achieved by executing tasks in a static partition of disjoint time slots.

To characterize a reservation independently on the specific implementation, Mok et al. [6] introduced the concept of *bounded delay partition* that describes a reservation R_k by two parameters: a bandwidth α_k and a delay Δ_k. The bandwidth α_k measures the fraction of resource that is assigned to the served tasks, whereas the delay Δ_k represents the longest interval of time in which the resource is not available. In general, the minimum service provided by a resource can be precisely described by its *supply function* [7], [8], representing the minimum amount of processing time the resource can provide in a given time interval.

Definition 1. *Given a reservation, the* supply function $Z_k(t)$ *is the minimum amount of time provided by the reservation in every time interval of length* $t \geq 0$.

In the particular case in which a reservation is implemented by a periodic server with unspecified priority that allocates a budget Q_k every period P_k, then the supply function is the one illustrated in Figure 2, where

$$\begin{cases} \alpha_k &= Q_k/P_k \\ \Delta_k &= 2(P_k - Q_k). \end{cases} \tag{6}$$

Once the bandwidth and the delay are computed, the supply function of a resource reservation can be lower bounded by the following *supply bound function*:

$$\mathsf{sbf}_k(t) \stackrel{\text{def}}{=} \max\{0,\ \alpha_k(t - \Delta_k)\}. \tag{7}$$

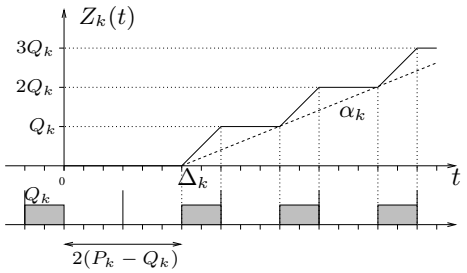


Fig. 2. A reservation implemented by a periodic server.

The supply bound function provides a nice abstraction for modeling a processor reservation R_k, because it is independent of the specific implementation and it allows characterizing the resource availability by only two parameters: the bandwidth α_k, which represents the fraction of the allocated resource, and the delay Δ_k, which represents the worst-case delay for using the resource. Figure 3 presents an alternative view of a processor consisting of two reservations dedicated to two tasks, τ_1 and τ_2. The (α_k, Δ_k) parameters of each reservation can be computed off-line to satisfy the computational demand of the served tasks, as presented by Buttazzo, Bini, and Wu [9].

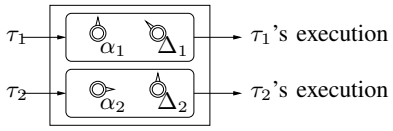


Fig. 3. An example of two reservations characterized by given bandwidth and delay parameters.

It is worth observing that reservations with smaller delays are able to serve tasks with shorter deadlines, providing better responsiveness. However, small delays can only be achieved with servers with a small period, which introduce more runtime overhead due to preemptions. If σ is the runtime overhead due to a context switch (subtracted from the budget every period), then the effective bandwidth of reservation R_k is

$$\alpha_k^{\mathsf{eff}} = \frac{Q_k - \sigma}{P_k} = \alpha_k \left(1 - \frac{\sigma}{Q_k}\right).$$

Expressing Q_k and P_k as a function of α_k and Δ_k we have

$$\begin{aligned} Q_k &= \frac{\alpha_k \Delta_k}{2(1-\alpha_k)} \\ P_k &= \frac{\Delta_k}{2(1-\alpha_k)}. \end{aligned}$$

Hence,

$$\alpha_k^{\mathsf{eff}} = \alpha_k + \frac{2\sigma(1-\alpha_k)}{\Delta_k}. \tag{8}$$

Within a reservation, the schedulability analysis of a task set under fixed priorities can be performed through the following Theorem [10]:

Theorem 1 (Bini et al., 2009)**.** *A set of preemptive periodic tasks with relative deadlines less than or equal to periods can be scheduled by a fixed priority algorithm, under a reservation characterized by a supply function* $Z_k(t)$*, if and only if*

$$\forall i = 1, \ldots, n \quad \exists t \in (0, D_i] : W_i(t) \leq Z_k(t), \tag{9}$$

where $W_i(t)$ *represents the Level-i workload, computed as follows:*

$$W_i(t) = C_i + \sum_{h:P_h > P_i} \left\lceil \frac{t}{T_h} \right\rceil C_h. \tag{10}$$

Similarly, the schedulability analysis of a task set under EDF can be performed using the following theorem [10]:

Theorem 2 (Bini et al., 2009)**.** *A set of preemptive periodic tasks with utilization U_p and relative deadlines less than or equal to periods can be scheduled by EDF, under a reservation characterized by a supply function $Z_k(t)$, if and only if $U_p < \alpha_k$ and*

$$\forall t > 0 \quad \mathsf{dbf}(t) \leq Z_k(t), \tag{11}$$

where $\mathsf{dbf}(t)$ is the Demand Bound Function [11] defined as

$$\mathsf{dbf}(t) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i. \tag{12}$$

Note that, if $Z_k(t)$ is lower bounded by the supply bound function, the test becomes only sufficient.

IV. Adapting to dynamic loads

Although resource reservation is essential for achieving predictability and isolation in the presence of tasks with variable execution times, the overall system performance strongly depends on a correct bandwidth allocation. In fact, if the processor bandwidth reserved to a task is much less than its average computational demand, the task may slow down too much, degrading the system's performance. On the other hand, if the allocated bandwidth is much greater than the actual needs, the system will run with low efficiency, wasting the available resources.

Unfortunately, a precise off-line evaluation of the reservation parameters is not always possible for different reasons. From one hand, the estimation of the computational requirements of a task is data dependent and is affected by several architecture features, such as cache, pre-fetch mechanism, and device access policies. On the other hand, a task may enter different operational modes at runtime, so the amount of resource reserved for a mode may not be suitable in another mode.

To cope with such a dynamic behavior, whenever the reserved bandwidth does not match the actual computational demand, the system should be able to adapt the reservation parameters to better satisfy the application needs. The adaptation scheme used in this paper makes use of a global reservation manager, which sense the actual resource usage through proper kernel probes and adapts the reservation parameters to better satisfy the application demand, while ensuring the schedulability of the overall system.

Reservation parameters can be changed upon specific requests coming from the served tasks, or directly by the manager itself, which periodically monitors the performance of the application and reallocate the resources to better meet the actual computational demand. Figure 4 shows a typical example of a global adaptation scheme, where a resource manager receives the actual resource consumption as a feedback from the kernel and tunes the reservation parameters to match the actual demand.

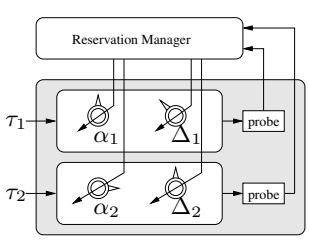


Fig. 4. Adaptive scheme adopted by a global Reservation Manager.

Providing the actual resource consumption of each task τ_i requires the operating system to have a specific monitoring mechanisms (kernel probes) to keep track of the actual job execution times and response times. Given these measurements, the actual computational demand $\hat{C}_i$ of a task τ_i can be estimated by averaging the actual computation times in a moving window containing the history of the last M measurements $e_{i,k}, e_{i,k-1}, \ldots, e_{i,k-M+1}$, where k denotes the current job index. This value can be used to evaluate the bandwidth currently needed by the task, as $\hat{U}_i = \hat{C}_i/T_i$. Then, $\hat{U}_i$ can be used as a reference value in a feedback loop to adapt the reservation bandwidth allocated to the task according to the actual needs. Note that, whenever a reservation is adapted online, the resource manager must also guarantee the schedulability of each reservation by ensuring that the overall allocated bandwidth does not exceed the maximum utilization bound of the adopted scheduling algorithm.

If the sum of all reservation bandwidths exceeds the maximum utilization bound, then the resource manager must apply a compression algorithm to bring the overall allocated bandwidth below such a bound.

1) Bandwidth compression: A simple ad efficient method for compressing a set of utilizations up to a given desired value has been proposed by Buttazzo et al. [12], and later extended by the same authors to deal with shared resources [13]. The idea is to consider each reservation as flexible as a spring with a given elasticity $E_i \geq 0$, whose bandwidth must be within a given range $[\alpha_i^{min}, \alpha_i^{max}]$. The elastic coefficient specifies the flexibility of the reservation to vary its bandwidth for adapting the system to a new feasible configuration. The greater E_i, the more elastic the reservation.

Under the elastic model, a set of n reservations with total utilization $U > 1$, can be compressed to reach a new desired utilization $U_d \leq 1$ such that all the bandwidths are within their ranges. Let us define $U_{min} \stackrel{def}{=} \sum_{i=1}^{n} \alpha_i^{min}$ and $U_{max} \stackrel{def}{=} \sum_{i=1}^{n} \alpha_i^{max}$. Then, if $U_{min} \leq U_d$, a feasible solution can always be found; hence, this condition has to be verified a priori.

In the special case in which $\alpha_i^{min} = 0$ for all the reservations, the compressed bandwidths can be derived by solving a set of n spring linear equations, under the constraint that $\sum_{i=1}^{n} \alpha_i = U_d$. The resulting compressed values are:

$$\forall i \quad \alpha_i = \alpha_i^{max} - (U_{max} - U_d)\frac{E_i}{E_s}, \tag{13}$$

where $E_s = \sum_{i=1}^{n} E_i$.

If $\alpha_i^{min} > 0$, the solution requires an iterative process. In fact, if during compression one or more bandwidths reach their minimum value, the additional compression can only vary the remaining reservations. Thus, at each instant, the set Γ of reservations can be divided into two subsets: a set Γ_f of fixed ones having minimum bandwidth, and a set Γ_v of variable ones that can still be compressed. If U_v^{max} is the sum of the maximum bandwidths in Γ_v, and U_f is the total utilization in Γ_f, then, to achieve a desired utilization $U_d \leq 1$, each reservation has to be compressed according to the following equation:

$$\forall \tau_i \in \Gamma_v \quad \alpha_i = \alpha_i^{max} - (U_v^{max} - U_d + U_f)\frac{E_i}{E_v}, \tag{14}$$

where $E_v = \sum_{\tau_i \in \Gamma_v} E_i$.

If there are reservations for which $\alpha_i < \alpha_i^{min}$, then α_i has to be fixed at α_i^{min}, sets Γ_f and Γ_v must be updated (hence, U_f and E_v recomputed), and Equation (14) applied again to the reservations in Γ_v. If there is a feasible solution, that is, if $U_{min} \leq U_d$, the iterative process ends when each value

α_i computed by Equation (14) is greater than or equal to its corresponding minimum α_i^{min}.

V. Schedulability Analysis

This section defines different probabilistic tests for analyzing the schedulability of a real-time system with adaptive reservations. By making use of probabilistic models, with their flexible system behavior representation, we aim at less pessimistic results to system schedulability. Furthermore, with probabilities we are able to provide a more fine grained evaluation of system timing performance i.e., respecting execution deadlines, in case of system changes, with soft and hard real-time constraints, as well as mixed-criticalities. All of that is achievable with different confidence/probabilities.

At the global level, the schedulability test has to verify that the sum of the bandwidths allocated to each application do not exceed the total bandwidth provided by the resource, that is $\sum_i \mathsf{sbf}_i \leq t$. This is equivalent to verify that $max\{0, \alpha_1(t - \Delta_1)\} + max\{0, \alpha_2(t - \Delta_2)\} + \ldots + max\{0, \alpha_n(t - \Delta_n)\} \leq t$. In terms of reservation bandwidths, it must be that $\sum_j U_j \leq 1$, where U_j is the utilization bandwidth allocated to the j-th partition[3].

At the local level, the schedulability test has to verify that, in any interval of time, the resource requested by the application does not exceed the available resource. This work considers EDF as a local scheduler, although similar reasoning applies for fixed priority (FP) scheduling.

We consider the probabilistic case with input pWCET and pMIT for tasks, as in Equation (2) and Equation (3). The probabilities are worst-case distributions and they are assumed to be given. Previous works have showed how to derive worst-case distributions from measurement-based probabilistic timing analyses, [14], [15]. The same approach with measurements and statistical analyses can be applied to derive pMITs.

In this paper, adaptivity is considered only at the level of reservations, which can be varied according to different system conditions. The analysis of the systems including adaptive applications will be addressed in a future work.

The probabilistic schedulability analysis has been addressed by Abeni, Manica, and Palopoli [16], who proposed the use of robust probabilistic guarantees, or by Maxim and Cucu-Grosjean [17], who derived a probabilistic response time analysis of real-time tasks under FP scheduling. Both approaches, however, are difficult to apply and are characterized by high complexity. The method proposed in this paper provides an efficient probabilistic analysis valid for adaptive conditions, while evaluating the effects of reservations changes on the application schedulability.

Example 1. *Let* $\Gamma = \{\tau_1, \tau_2\}$ *be the task set of an application running under a resource partition, where* $\tau_1 = (\mathcal{C}_1, \mathcal{T}_1, D_1)$ *is a probabilistic periodic task with* $D_1 = 8$, $\mathcal{C}_1 = \begin{pmatrix} 2 & 3 & 4 \\ 0.5 & 0.4 & 0.1 \end{pmatrix}$ *and* $\mathcal{T}_1 = \begin{pmatrix} 12 & 10 & 8 \\ 0.1 & 0.2 & 0.7 \end{pmatrix}$, *and* $\tau_2 = (\mathcal{C}_2, T_2, D_2)$ *is another probabilistic periodic task with* $T_2 = 10$, $D_2 = 10$ *and* $\mathcal{C}_2 = \begin{pmatrix} 2 & 3 & 5 & 6 \\ 0.1 & 0.4 & 0.5 & 0.1 \end{pmatrix}$. *The set* Γ *is used in the rest of this section for explaining the schedulability effects resulting from adaptive reservations. From now on, time is considered to be discrete and will be expressed in ticks, although conclusions can be made with respect to any time unit.*

[3]The available resource is fixed at $\mathsf{sbf} = t$ or bandwidth $U = 1$ in case of single processor. In case of multiple processors the available resource is larger, then the schedulability conditions change.

A. Probabilistic Behavior Bounding

The first analysis we propose focuses on supply bound functions sbf and demand bound functions dbf for representing the reservations and the applications, respectively.

Probabilistic demand bound functions can be defines using pWCETs and pMITs as sets of probabilistic bounds of the task execution behavior. Since each combination of worst-case execution time and minimum inter-arrival time (C^k, T^l) defines a specific periodic task behavior, there exists a probability associated with the couple (C^k, T^l), representing the probability for that combination to happen. Hence, from (C^k, T^l) it is possible to derive the demand bound function (dbf, p) with an associated probability p, representing the probability that the resource demand of τ_i is upper bounded by dbf. The whole set of demand bound functions and probabilities (one (dbf, p) per (C^k, T^l) combination) forms a distribution of demand bound functions $\langle \mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x) \rangle$, called *probabilistic demand bound function*. This function can also be represented as follows, by making use of a parameter x:

- $\mathsf{dbf}(t, x)$ is the demand bound function corresponding to a value of x;
- $F_{\mathsf{dbf}}(x)$ is the probability that the resource demand of the task set is upper bounded by $\mathsf{dbf}(t, x)$. Such a probability results from the CDF of the input $\mathcal{C}$ and/or $\mathcal{T}$ distributions.

The probabilistic demand bound function $\langle \mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x) \rangle$ is inspired by [18], [19], and it comes with the rationale that we randomly pick values from $\mathcal{C}$ and $\mathcal{T}$ once, at the beginning of the task execution. From each of those picks we build $(\mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x))$. Figure 5 depicts the demand bound curves for different values of x: the accuracy of the bound $\mathsf{dbf}(x)$ increases with x, as the probability that $\mathsf{dbf}(x)$ upper bounds task resource demand increases. $(\mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x) = 1)$ is the deterministic case, where dbf upper bounds task resource demand 100% of the times. the probabilistic demand bound

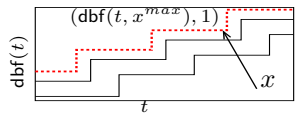


Fig. 5. Probabilistic demand bound function from ordered bounds.

function for τ_i, The probabilistic demand bound function of the application $\langle \mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x) \rangle$ is defined by composing the probabilistic demand bound functions $\langle \mathsf{dbf}_i(t, x), F_{\mathsf{dbf}_i}(x) \rangle$ of each task τ_i. The composition, as convolution of the distributions $\langle \mathsf{dbf}_i(t, x), F_{\mathsf{dbf}_i}(x) \rangle$, is guaranteed by the assumed independence of tasks.

1) dbf *Probabilistic Schedulability:* For the local schedulability analysis with probabilistic demand bound functions we extend Theorem 2 to the probabilistic case by making use of $\langle \mathsf{dbf}(t, x), F_{\mathsf{dbf}}(x) \rangle$ and the reservation sbf.

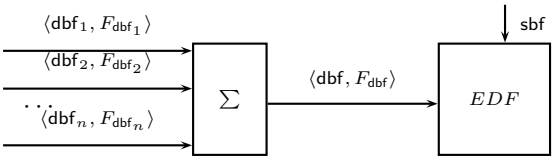


Fig. 6. An example of EDF scheduling with probabilistic demand bound functions and supply bound function.

Although $p = F_{\mathsf{dbf}}(x)$ describes the probability that $\mathsf{dbf}(t,x)$ upper bounds the application execution behavior (e.g. $\mathsf{dbf}(t,x)$ being an upper bound), that probability can be translated into schedulability probability by comparing dbf and sbf. From Equation (11), if dbf $>$ sbf there will be a deadline miss, thus the system is not schedulable. With $(\mathsf{dbf}(t,x), p = F_{\mathsf{dbf}}(x))$, $1-p$ is the probability of exceeding dbf, thus the probability that the condition dbf $\leq$ sbf is not respected.

Theorem 3 (1-EDF Schedulability)**.** *Any probabilistic task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ *with* $\langle \mathsf{dbf}, F_{\mathsf{dbf}} \rangle$ *is EDF schedulable with a probability level* 1 *upon a resource provisioning curve* sbf *if*

$$\forall\, t > 0,\ \mathsf{dbf}(t) \leq \mathsf{sbf}(t). \tag{15}$$

Here, dbf *represents the upper bounding of* $\langle \mathsf{dbf}, F_{\mathsf{dbf}} \rangle$ *and* sbf *the resource provisioning curve.*

Proof: The proof of this theorem follows directly from the fact that both dbf and sbf are bounding curves with a probability 1 each. The theorem follows. ∎

Theorem 15 states the deterministic schedulability with reservation sbf.

Theorem 4 (p-EDF Schedulability)**.** *Any probabilistic task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ *resulting in a probabilistic resource demand* $\langle \mathsf{dbf}, F_{\mathsf{dbf}} \rangle$ *is schedulable under EDF with a probability level* $p \in [0,1[$ *and upon a resource curve* sbf *if* $\forall t > 0$,

$$\exists\, x_1 \geq 0 \ \textit{ such that } \mathsf{dbf}(t,x_1) \leq \mathsf{sbf}(t) \textit{ and } F_{\mathsf{dbf}}(x_1) \geq p. \tag{16}$$

Proof: The schedulability is guaranteed when the request is less than the provisioning. Accordingly, to ensure a schedulability with a certain probability level, it is sufficient to find a request curve $\mathsf{dbf}(\Delta, x_1)$ upper bounded by sbf. Since $\mathsf{dbf}(t,x)$ and sbf are independent random variables[4] and $F_{\mathsf{dbf}}(x_1)$ is the probability that $\mathsf{dbf}(t,x_1)$ upper bounds the request function, then $F_{\mathsf{dbf}}(x_1)$ is the probability that $\mathsf{dbf}(t,x_1)$ upper bounds the request function on one hand, and $\mathsf{sbf}(t)$ lower bounds the resource provisioning on the other hand. As such, $1 - F_{\mathsf{dbf}}(x_1)$ is the probability that the events are not bounded by the two functions, i.e., the probability that the events are not schedulable by the resource provisioning sbf. Consequently, a schedulability condition with a probability level p-EDF schedulability is guaranteed whenever $\mathsf{dbf}(t,x_1) \leq \mathsf{sbf}(t)$ and $F_{\mathsf{dbf}}(x_1) \geq p$. The theorem follows. ∎

Theorem 4 states the probabilistic schedulability with reservation sbf.

Example 2. *Considering* Γ *from Example 1. For* τ_1*, the probabilistic demand bound function is represented in Figure 7. In particular, the first part shows all the possible curves* $(\mathsf{dbf}(t,x), F_{\mathsf{dbf}}(x))$*, while the second one shows a subset of those curves reduced to just the meaningful ones. Partial ordering and equivalence between curves is stated in [18] and similar to the partial ordering between distributions in [20]. In [18] it is also described the strategy to reduce the number of curves (thus the complexity) to the non overlapping ones by dominating curves. Figure 8 describes the pdf* $f_{\mathsf{dbf}_1}(x)$ *and CDF* $F_{\mathsf{dbf}_1}(x)$ *of the reduced set of curves, each with an index associated.*

The probabilistic approach provides more flexibility than deterministic models/analyses (probabilistic schedulability with $p = 1$ all the time) by adding more intermediate possible schedulability conditions with $p \leq 1$.

[4]The supply bound function could be seen as a random variable with just one value.

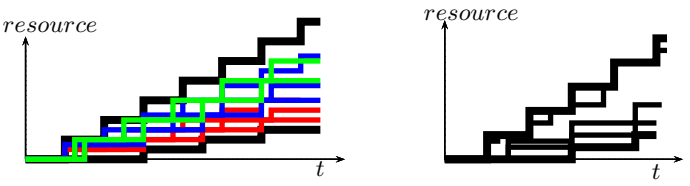


Fig. 7. Probabilistic curve: complete set of ordered curves $\mathsf{dbf}_1(t,x)$ and the reduced set of probabilistic bounds.

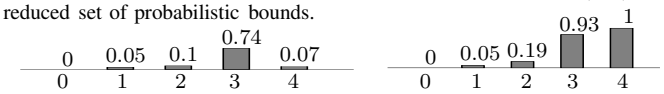


Fig. 8. Probabilistic curve: distribution function $f_{\mathsf{dbf}_1}(x)$ and cumulative distribution function $F_{\mathsf{dbf}_1}(x)$ indexed by x.

2) Adaptation Effects: Reservation changes could affect components/partitions schedulability. In the case of probabilistic schedulability, the effect is quantified by the schedulability probability.

By applying sbf_1 the schedulability probability could result into p_1, from Theorem 4. Then, changing the reservation to sbf_2, the schedulability probability could become p_2, depending on the largest $(\mathsf{dbf}(t,x_i), p_i = F_{\mathsf{dbf}}(x_i))$ such that $\mathsf{dbf}(t,x_i) \leq \mathsf{sbf}_2(t)$. Thus, for any given t, $p_2 = max_i\{p_i \mid \mathsf{dbf}(t,x_i) \leq \mathsf{sbf}_2(t)\}$.

The effect of changes reflects on the probabilities, with $p_2 - p_1$ being the probability variation for the application. Such effects can be visually depicted in an (α, Δ)-space, having reservation bandwidth and service delay on the axes. A point (α_i, Δ_i) in such a plain represents a supply bound function sbf. In a deterministic framework, the schedulability region in the (α, Δ)-space has been described by Lipari and Bini [7]. In this work, this region is extended to incorporate probabilities.

A probabilistic schedulability region of a $(\mathsf{dbf}(t,x), p = F_{\mathsf{dbf}}(x))$ in the probabilistic (α, Δ)-space is the set of tuples (α_i, Δ_i) that makes the dbf probabilistically schedulable with a probability larger than or equal to p. A change in the supply bound function corresponds to a change in the schedulability region and in the related probabilities, and affects system schedulability performance.

Example 3. *Let us consider* Γ *from Example 1 and three reservations* $\mathsf{sbf}_1 = (0.98, 0.4)$*,* $\mathsf{sbf}_2 = (0.86, 3)$*, and* $\mathsf{sbf}_3 = (0.8, 3)$*. Figure 9 describes three different EDF schedulability regions obtained from 3 curves shown in Example 2. In particular,*

- $\mathsf{dbf}_a = \mathsf{dbf}_1(t,x_1) + \mathsf{dbf}_2(t,x_{2,a})$ *with* $x_{2,a}$ *such that* $\mathsf{dbf}_2(t,x_{2,b}) = \mathsf{dbf}_{2,C_2=2}$ *and* $p_{2,a} = 0.1$*;*
- $\mathsf{dbf}_b = \mathsf{dbf}_1(t,x_1) + \mathsf{dbf}_2(t,x_{2,b})$ *with* $x_{2,b}$ *such that* $\mathsf{dbf}_2(t,x_{2,b}) = \mathsf{dbf}_{2,C_2=3}$ *and* $p_{2,b} = 0.4$*;*
- $\mathsf{dbf}_c = \mathsf{dbf}_1(t,x_1) + \mathsf{dbf}_2(t,x_{2,c})$ *with* $x_{2,c}$ *such that* $\mathsf{dbf}_2(t,x_{2,c}) = \mathsf{dbf}_{2,C_2=4}$ *and* $p_{2,c} = 0.9$*.*

x_1 *is such that* $\mathsf{dbf}_1(t,x_1 = max\{\mathsf{dbf}_{1,(C_1=2,T_1=8)}, \mathsf{dbf}_{1,(C_1=3,T_1=8)}, \mathsf{dbf}_{1,(C_1=3,T_1=10)}, \mathsf{dbf}_{1,(C_1=4,T_1=12)}, \mathsf{dbf}_{1,(C_1=4,T_1=12)}\}$*.* $p_1 = 0.93$ *as from Figure 8 curves* 3*. Each region has a probability associated,* $p_a = 0.93 * 0.1 = 0.093$*,* $p_b = 0.93 * 0.4 = 0.372$*,* $p_c = 0.93 * 0.9 = 0.837$*. From the probabilistic* (α, Δ)*-space, we conclude what follows.*

- sbf_1 *makes feasible all the* dbf*, then the maximum schedulability probability that guarantees is* $max\{p_a, p_b, p_c\}$*.*
- sbf_2 *makes feasible* dbf_a *and* dbf_b*, and its maximum schedulability probability is* p_b*.*

- sbf_3 *makes feasible* dbf_a *only. Since* sbf_3 *is not on the boundary of the region, we cannot conclude on the exact schedulability probability. Nonetheless, the schedulability probability with reservations* sbf_3 *is larger than* 0.093 *and smaller than* 0.373.

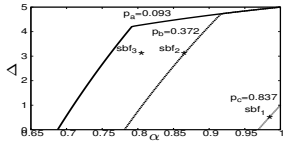


Fig. 9. The probabilistic (α, Δ)-space with feasibility regions, probabilities associated, and reservations.

The $\langle \mathsf{dbf}(t,x), F_{\mathsf{dbf}}(x) \rangle$ and the probabilistic (α, Δ)-space could be also applied for reservation design depending on the deadline miss constraints of each system component.

B. Cumulative Probabilistic Processes Bounding

Using probabilistic models, the resource demand of a task τ_i in an specific time interval can be seen as a cumulative probabilistic process where, at each task instance, all the possible combinations for pWCET and pMIT are considered. $\mathcal{DBF}_i(t)$ is the cumulative probabilistic demand bound function that τ_i could demand in $[0,t]$; $\mathcal{DBF}_i(t) = \otimes^{\eta}\mathcal{C}_i$, where η is the number of $\mathcal{C}_i$ instances to be considered in $[0,t]$. $\mathcal{DBF}_i$ is obtained by randomly picking a task worst-case execution time from $\mathcal{C}_i$ and a minimum inter-arrival time from $\mathcal{T}_i$ at each task instance in the time interval $[0,t]$. With both pWCET and pMIT, the τ_i cumulative probabilistic demand bound function is

$$\mathcal{DBF}_i(t) = \otimes_{\lceil \frac{t+\mathcal{T}_i-d_i}{\mathcal{T}_i} \rceil} \mathcal{C}_i, \tag{17}$$

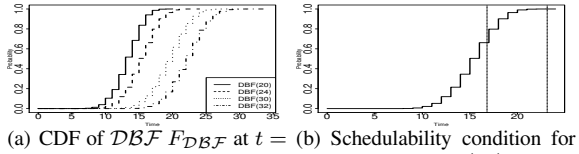
where $\lceil \frac{t+\mathcal{T}_i-d_i}{\mathcal{T}_i} \rceil$ defines the number of convolutions of $\mathcal{C}_i$. $\lceil \frac{t+\mathcal{T}_i-d_i}{\mathcal{T}_i} \rceil$ is a distribution where the values are the number of $\mathcal{C}_i$ instances to account for, and the probabilities are the chances that that number of instances happens within $[0,t]$. For the whole application task set the cumulative probabilistic demand bound function is $\mathcal{DBF}(t) = \otimes_i \mathcal{DBF}_i(t)$. The composition by convolution (summing up random variables $\mathcal{C}$) is allowed by the hypothesis of independence between tasks and task instances we have made.

1) $\mathcal{DBF}$ Probabilistic Schedulability: For an interval $[0,t]$, a task application, with probabilistic resource demand as $\mathcal{DBF}(t)$ and served by a reservation $\mathsf{sbf}(t)$, is schedulable with probability p if $\mathbb{P}\{\mathcal{DBF}(t) \leq \mathsf{sbf}(t)\} \geq p$. The probability of deadline miss at time t (probability of not schedulability in $[0,t]$) is then $1-p$.

In this case, we cannot compose EDF schedulability conditions as done in Theorem 2 and Theorem 4. This is due to the dependence existing between two intervals $[0,t_1]$ and $[0,t_2]$ for the conditions $\mathcal{DBF}(t_1) \geq \mathsf{sbf}(t_1)$ and $\mathcal{DBF}(t_2) \geq \mathsf{sbf}(t_2)$: the two intervals overlap (being $t_1 < t_2$) making the two conditions statistically dependent. In order to compute the joint probability $P\{\mathcal{DBF}(t_1) \leq \mathsf{sbf}(t_1), \mathcal{DBF}(t_2) \leq \mathsf{sbf}(t_2)\} = P\{\bigwedge_{t_i \in \{t_1,t_2\}} \mathcal{DBF}(t) \leq \mathsf{sbf}(t)\}$, such a dependence needs to be characterized. Since in this paper such a dependence is not fully modeled, the EDF probabilistic schedulability analysis with $\mathcal{DBF}$ is left to future work. However, the partial schedulability conditions (single intervals) derived above can be used to characterize the effects of a reservation change for each interval $[0,t]$. It is worth noting that in case of Theorem 16 the statistical independence between two conditions $\mathsf{dbf}(t_1,x) \leq \mathsf{sbf}(t_1)$ and $\mathsf{dbf}(t_2,x) \leq \mathsf{sbf}(t_2)$ is guaranteed because we have randomly extracted values only once and at the beginning of task executions.

With a reservation sbf_1 it is $p_1 = \mathbb{P}\{\mathcal{DBF}(t) \leq \mathsf{sbf}_1(t)\}$, and due to the change to sbf_2, that probability changes to p_2 where $p_2 = \mathbb{P}\{\mathcal{DBF}(t) \leq \mathsf{sbf}_2(t)\}$. The schedulability changes within $[0,t]$ is quantified by $p_2 - p_1$.

Example 4. *Given* Γ *from Example 1, and the reservations* $\mathsf{sbf}_1(t)$ *and* $\mathsf{sbf}_3(t)$ *from Example 3, the effects of a change from* $\mathsf{sbf}_1(t)$ *to* $\mathsf{sbf}_3(t)$ *are depicted in Figure 10. Figure 10(a) describes the different cumulative probabilistic demand bound functions for* $t = 20$, $t = 24$, $t = 30$ *and* $t = 32$. *From the figure it is possible to outline the difference between the demands due to task behaviors and the length of the analysis intervals. Noticeable is the dominance of* $\mathcal{DBF}(32)$ *over all the others:* $\mathcal{DBF}(32)$ *is larger than the other cumulative demand distributions*[5]. *Figure 10(b), with the CDF representation of* $\mathcal{DBF}(24)$, *shows the difference in terms of probability due to the changes from* $\mathsf{sbf}_1(24)$ *to* $\mathsf{sbf}_3(24)$. *The difference is such that the schedulability probability changes from* $p_1 = \mathbb{P}\{\mathcal{DBF}(t) \leq \mathsf{sbf}_1(t)\} = 0.999993$ *to* $p_3 = \mathbb{P}\{\mathcal{DBF}(t) \leq \mathsf{sbf}_3(t)\} = 0.661624$. *Hence, the reservation change affects* Γ *schedulability,*



(a) CDF of $\mathcal{DBF}$ $F_{\mathcal{DBF}}$ at $t = 20$, $t = 24$, $t = 30$ and $t = 32$

(b) Schedulability condition for at $t = 24$ with $\mathsf{sbf}_1(24) = 16.8$ and $\mathsf{sbf}_3(24) = 23.128$

Fig. 10. CDF representation and single interval schedulability condition $\mathcal{DBF}(t) \leq \mathsf{sbf}(t)$ for $t = 24$.

reducing the probability by $|p_2 - p_1| = 0.3383753$. *In order to achieve schedulability probability* $p = 1$ *in* $[0, 24]$ *it is necessary a reservation with* $\mathsf{sbf}(24) \geq 24$, *thus* $\mathsf{sbf} = (1, 0)$.

The schedulability analysis with $\mathcal{DBF}$ provides a more accurate representation of the schedulability conditions than with dbfs, and a better characterization of the effects of changing reservations than with dbfs. This is due to the larger amount of system behaviors accounted for by the $\mathcal{DBF}$.

C. Probabilistic Utilization

From the utilization perspective, in order to guarantee probabilistic schedulability of the reservation it is sufficient to compare the application utilization discrete distribution $\overline{\mathcal{U}}$ with the reservation bandwidth threshold U.

For an application characterized by $\overline{\mathcal{U}}$, the resource reservation bandwidth U fixes the limit for the application utilization in order to be schedulable. Hence, the schedulability probability p is such that

$$p = \mathbb{P}\{\overline{\mathcal{U}} \leq U\}. \tag{18}$$

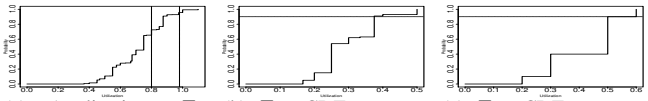
Analyzing the reservation bandwidth change from U_1 to U_2, schedulability effects are quantified by the difference between $p_1 = \mathbb{P}\{\overline{\mathcal{U}} \leq U_1\}$ and $p_2 = \mathbb{P}\{\overline{\mathcal{U}} \leq U_2\}$.

Example 5. *Considering* Γ *from Example 1, the application utilization* $\overline{\mathcal{U}}$ *is depicted as the CDF in Figure 11(a). In there, also* p_3 *and* p_1 *are represented when the reservation changes from* $U_3 = 0.8$ *to* $U_1 = 0.98$. U_3 *and* U_1 *from Example 3. The change from* U_3 *to* U_1 *increases* Γ *schedulability probability by* $|0.725 - 0.956| = 0.231$.

Figure 11 shows the possibility of evaluating the impact of an assumed schedulability probability of 0.9 *per task. Concerning*

[5]The partial ordering between distributions is defined according to [20], where, given two distributions $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{X}$ is greater than or equal to $\mathcal{Y}$ ($\mathcal{X} \succeq \mathcal{Y}$) if and only if $\forall x\ \mathbb{P}\{\mathcal{X} \leq x\} \leq \mathbb{P}\{\mathcal{Y} \leq x\}$.

τ_1, *all the combinations of* $\mathcal{C}_1$ *and* $\mathcal{T}_1$ *are limited to those for which* $F_{\overline{\mathcal{U}}_1}(x) \leq 0.9$ *(*τ_1 *utilization limited to* 0.333*).* τ_2 *is limited to worst-case execution time* $C_2 \leq 5$ *which give an upper bound to* τ_2 *utilization of* 0.5.



(a) Application reservation utilizations at 0.98 and 0.8. (b) $F_{\overline{\mathcal{U}}_1}$ CDF representation for τ_1 (c) $F_{\overline{\mathcal{U}}_2}$ CDF representation for τ_2

Fig. 11. Task utilization bounding at probability $p = 0.9$.

As shown in Example 5, with the utilization approach it is possible to relate schedulability probabilities to task parameters, setting them to achieve a defined schedulability goal. In a future work we plan to exploit the utilization probabilistic schedulability to design the task set parameters in order to maximize a given probabilistic performance index.

VI. Conclusions

The current trend in embedded systems shows that hardware is evolving more rapidly than software, causing a strong need for methodologies able to achieve portability, modularity, and scalability of performance. Surprisingly, such a growth in hardware complexity was not balanced by a corresponding evolution of the control software for a predictable an efficient management of the computational resources. As a consequence, most of today's embedded systems are still implemented on top of fixed priority kernels or by adopting ad hoc techniques to exploit new hardware features.

This chapter presented some new software methodologies proposed within the real-time research community to simultaneously address multiple objectives, such as predictability, efficiency, modularity, portability, and adaptability.

Predictability and efficiency is achieved through the use of proper real-time scheduling algorithms (like Rate Monotonic and Earliest Deadline First), which optimally utilize the processor and can be efficiently analyzed to estimate the worst-case response time of each task. Modularity and portability is obtained through the use of resource reservation mechanisms, which allow implementing a temporal protection environment. In this way, a task can be analyzed independently of the behavior of the other tasks, but only as a function of its computational requirements and the allocated bandwidth. Finally, adaptability is achieved through the use of proper feedback schemes that can be implemented both at the application level (local adaptation) or at the system level (global adaptation).

The probabilistic analyses are for the first time applied to the adaptivity problem, and have proved flexibility and accuracy to proficiently validate safety of real-time systems.

However, the real challenge for the future is to transfer such techniques to the industry to make next generation embedded systems more predictable and efficient, as well as portable to different computing platforms, and adaptable to dynamic load conditions.

Future work will address mainly the probabilistic analysis and its coupling with the adaptive resource reservation mechanisms. Predictability and adaptivity promptness trade-offs will be investigated for developing optimal and sub-optimal adaptive systems. Furthermore, accuracy and complexity trade-offs of probabilistic schedulability analysis will be approached with the help of re-sampling techniques, [21]. Finally, the mixed-criticality approach will be combined with adaptive systems to provide a complete perspective to aerospace embedded systems.

References

[1] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," in *Proceedings of IEEE international conference on Multimedia Computing and System*, May 1994.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, 1973.

[3] L. Abeni and G. C. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS'98)*, Madrid, Spain, December 2-3, 1998.

[4] ——, "Resource reservations in dynamic real-time systems," *Real-Time Systems*, vol. 27, no. 2, pp. 123–165, 2004.

[5] D. Faggioli, F. Checconi, M. Trimarchi, and C. Scordino, "An edf scheduling class for the linux kernel," in *Proceedings of the 11th Real-Time Linux Workshop (RTLW 2009)*, Dresden, Germany, September 2009.

[6] A. K. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, Taipei, Taiwan, May 2001, pp. 75–84.

[7] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *Proceedings of the 15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, Porto, Portugal, July 2003, pp. 151–158.

[8] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th Real-Time Systems Symposium*, Cancun, Mexico, December 2003, pp. 2–13.

[9] G. C. Buttazzo, E. Bini, and Y. Wu, "Partitioning real-time applications over multicore reservations," *IEEE Trans. Industrial Informatics*, vol. 7, no. 2, pp. 302–315, 2011.

[10] E. Bini, G. C. Buttazzo, and G. Lipari, "Minimizing CPU energy in real-time systems with discrete speed management," *ACM Transactions on Embedded Computing Systems*, vol. 8, no. 4, pp. 31:1–31:23, July 2009.

[11] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Journal of Real-Time Systems*, vol. 2, 1990.

[12] G. C. Buttazzo, L. Abeni, and G. Lipari, "Elastic task model for adaptive rate control," in *IEEE Real Time System Symposium*, Madrid, Spain, December 1998.

[13] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, March 2002.

[14] J. Hansen, S. Hissam, and G. A. Moreno, "Statistical-Based WCET Estimation and Validation," in *9th International Workshop on Worst-Case Execution Time Analysis (WCET'09)*, ser. OpenAccess Series in Informatics (OASIcs), vol. 10, 2009, pp. 1–11.

[15] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F. J. Cazorla, "Measurement-Based Probabilistic Timing Analysis for Multi-path Programs," in *23nd Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2012.

[16] L. Abeni, N. Manica, and L. Palopoli, "Efficient and robust probabilistic guarantees for real-time tasks," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1147–1156, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2011.12.042

[17] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *Proceedings of the IEEE 34th Real-Time Systems Symposium, RTSS 2013, Vancouver, BC, Canada, December 3-6, 2013*, 2013, pp. 224–235.

[18] L. Santinelli and L. Cucu-Grosjean, "A probabilistic calculus for probabilistic real-time systems," *ACM Transactions on Embedded Computing Systems*, 2015.

[19] L. Santinelli, P. Meumeu, D. Maxim, and L. Cucu-Grosjean, "A component-based framework for modeling and analyzing probabilistic real-time systems," in *Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, 2011.

[20] J. Diaz, J. Lopez, M. Garcia, A. Campos, K. Kim, and L. L. Bello, "Pessimism in the stochastic analysis of real-time systems: Concept and applications," in *25th of the IEEE Real-Time Systems Symposium (RTSS)*, 2005.

[21] D. Maxim, M.Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean, "Re-sampling for statistical timing analysis of real-time systems," in *20th International Conference on Real-Time and Network Systems, RTNS*, 2012, pp. 111–120.