Stochastic Analysis of a Reservation Based System

Luca Abeni
RETIS Lab
Scuola Superiore S. Anna, Pisa
luca@sssup.it

Giorgio Buttazzo
University of Pavia (Italy)
INFM - Pavia research unit
giorgio@sssup.it

Abstract

Resource Reservation is an effective technique for allocating CPU time to concurrent real-time applications running on a uniprocessor system. The most important advantage of reservation-based CPU allocation is that it enforces temporal isolation, thus ensuring that the behaviour of each application will not depend on the temporal requirement of the others. In this paper a stochastic analysis of a generic reservation system is presented, enabling application designers to compute the probability of respecting deadlines when the probability distributions of the interarrival and execution times are known.

1. Introduction

Real-time theory was born in the early '70s to support control applications in which some activities must be executed periodically with a fixed period, and it is important to ensure that each periodic instance will finish before the end of its period. For this reason, classical real-time theory tends to assume fixed execution times and bounded interarrival times, guaranteeing that some temporal constraints, called deadlines, will always be respected.

However, in the last years real-time techniques resulted to be useful even in different application contexts, where the strict respect of every deadline is not necessary. In this case, *hard* real-time theory (that guarantees the respect of *all* the deadlines) can lead to an underutilization of the system, allocating "too much" resources to tasks.

In order to better utilize the system, a *soft* real-time approach can be used: a soft real-time task *should* respect its deadlines, but nothing dramatic happens if it misses a *reasonable number* of them. In this way, the task is no more required to respect all the deadlines, and the amount of resources dedicated to the task can be decreased achieving a better system utilization.

One of the biggest problems with soft real-time scheduling is to give a quantitative definition of the *reasonable number* of deadlines cited above: in this context we believe that a probabilistic description (in terms, for example, of deadline miss ratio) can help to describe a soft real-time system in a more rigorous way.

As said, there is an increasing interest in extending the use of real-time techniques to new fields: as an example, the integration of real-time activities [14] in an Open System [5, 4] or in a desktop Operating System is becoming an important topic in real-time research. Some interesting examples of this trend can be Linux/RK [18, 17], Nemesis [12], or Rialto [7].

In general, one of the most important properties to enforce is *temporal protection*, ensuring that the worst case temporal behaviour of a task will not be affected by the other tasks present in the system. In this way, a task requiring too much execution time cannot jeopardize the temporal guarantee of other correct-behaving tasks. The simplest way to enforce temporal protection is using resource reservation techniques [15, 18, 1], that add an enforcement strategy to classical real-time scheduling [14].

Under some assumptions (namely, if the worst case execution time and the minimum interarrival time of a task are known), it is possible to use the temporal isolation property to guarantee single tasks to respect their deadlines. However, it could be sometime useful to allow a task to miss some deadline in order to increase the execution rates of the tasks in the system [16]. In this case, if a resource reservation technique is used it is still possible to guarantee a minimum Quality of Service (QoS) for the tasks, in terms of probability of respecting a deadline.

In fact, in many time sensitive applications the requirements for a task are less stringent than classical real-time theory (based on the respect of every deadline of every task) usually permits. For example, in [19, 6], a probabilistic analysis of the Rate Monotonic (RM) scheduling algorithm is performed, and some solutions to guarantee a completion probability are presented.

In [3], the RM algorithm is modified to perform on-line acceptance of single jobs for ensuring (off line) a minimum probability that a job is accepted. Like the previous, this

paper is limited to a static priority (RM) assignment and to a semi-periodic (fixed interarrival times) task model.

In [8], a Time Division Multiplexing (TDM) model is used to share resources between tasks in a distributed system. In that paper a statistical approach is used at design phase to compute the probability distribution of tasks' finishing times and to check whether a distributed system verifies some given constraints on latency and minimum output rate.

In [9, 10, 11] a statistical analysis of the Earliest Deadline First (EDF) and other real-time scheduling algorithms is presented. That analysis is based on the Real-Time Queuing theory, which adapts the classical queuing theory to handle queues containing clients characterized by a deadline.

In [2], a similar analysis is performed for *generalized sporadic tasks* (fixed execution time and variable interarrival time) and *semi periodic tasks* (variable execution time and fixed interarrival time), when a Constant Bandwidth Server (CBS) [1] is used.

In this paper, we extend the approach presented in [2] applying a similar analysis to a generic reservation-based system, and releasing some assumptions. In particular, our new analysis is valid for variable execution times and variable interarrival times.

As in the previously presented works, we use the *completion probability* as a metric for expressing and guaranteeing a desired QoS for each task. While the aim of the classical hard schedulability analysis is to guarantee that each job will finish within its deadline, the aim of a QoS guarantee is to ensure that a job will finish within its soft deadline with a given probability. To perform a QoS guarantee we need to describe the tasks' parameters in a probabilistic way and introduce the concept of probabilistic deadline.

The rest of the paper is organized as follows. Section 2 presents the resource reservation abstraction and the task model used for our analysis, introducing some notations. Section 3 describes the analysis for performing a probabilistic guarantee. The presented analysis imposes strong constraints on the interarrival times; these constraints are relaxed in Section 4. Section 5 illustrates an example of the proposed analysis, while Section 6 presents our conclusions.

2. Scheduling Model

A real-time task τ_i is traditionally considered as a stream of instances, or jobs, $J_{i,j}$, each of them arriving at time $r_{i,j}$, executing for a time $c_{i,j}$ and then finishing at time $f_{i,j}$. Each job $J_{i,j}$ is also characterized by an absolute deadline $d_{i,j} = r_{i,j} + D_i$, being D_i the relative deadline of task τ_i.

If $r_{i,j} = r_{i,j-1} + T_i$, τ_i is said to be periodic, with period T_i, while if $r_{i,j} \geq r_{i,j-1} + T_i$ τ_i is sporadic and T_i is its Minimum Interarrival Time (MIT). If $D_i = T_i$, each task τ_i can be described by a pair (C_i, T_i), where $C_i = \max_j\{c_{i,j}\}$ is the Worst Case Execution Time.

Conventional real-time algorithms schedule tasks by period (Rate Monotonic) [14], by relative deadlines (Deadline Monotonic) [13], or by absolute deadlines (Earliest Deadline First) [14], and guarantee that **each job will finish within its absolute deadline** if an admission test is verified. The admission test is

$$\sum_i \frac{C_i}{T_i} \leq U_{lub} \tag{1}$$

where U_{lub} depends on the scheduling algorithm.

Reservation-based scheduling extends this approach by **enforcing the maximum execution time per period**. Each task is associated to two scheduling parameters (Q_i^S, T_i^S), meaning that τ_i will be reserved Q_i^S execution time units every T_i^S. If an admission test $\sum_i \frac{Q_i^S}{T_i^S} \leq U_{lub}$ is verified, each task is guaranteed to execute at least for the reserved amount of time.

Using this newer scheduling approach, it is possible to **separate the scheduling parameters** (Q_i^S, T_i^S) **from the task characteristics** (execution and interarrival times). In this way, a task τ_i can be described by a pair of Probability Distribution Functions (PDF):

$$\tau_i = (U_i(c), V_i(t))$$

where $U_i(c)$ is the probability that job $J_{i,j}$ has execution time c, and $V(t)$ is the probability that jobs' interarrival time is t. Hence,

$$\begin{aligned} U_i(c) &= P\{c_{i,j} = c\} \\ V_i(t) &= P\{r_{i,j+1} - r_{i,j} = t\}. \end{aligned}$$

It can be easily proved that if $\exists C_i : \forall x > C_i, U_i(x) = 0$ and $\exists T_i : \forall x < T_i, V_i(x) = 0$, assigning $Q_i^S > C_i$ and $T_i^S < T_i$, guarantees that τ_i will respect all its deadlines. This corresponds to the Liu and Layland priority assignment and analysis of Equation 1.

On the other hand, it could be interesting to assign the scheduling parameters (Q_i^S, T_i^S) to τ_i in a less conservative way, still maintaining some control on the QoS experienced by τ_i. We can use the concept of *probabilistic deadline* to quantify the QoS experienced by each task.

A probabilistic deadline δ is not required to be always respected, but can be respected by task τ_i with a probability

$$X_i(\delta) = P\{f_{i,j} \leq r_{i,j} + \delta\} < 1.$$

Performing a QoS guarantee using a probabilistic deadline δ means to guarantee that:

- if task τ_i is described by the PDFs $(U_i(c), V_i(t))$
- if the assigned scheduling parameters are (Q_i^S, T_i^S)
- then, each job of task τ_i has probability $X_i(\delta)$ of finishing within a relative deadline δ.

3. Schedulability Analysis

In this section we present our stochastic analysis of a generic reservation-based scheduling algorithm. In order to simplify the tractation, we consider a single task τ, removing the i index from all the symbols (hence, $U(c) = U_i(c)$, and so on).

Let $U(c)$ be a generic PDF for the task's computation times and let the interarrival times be multiple of T^S. In this case, $r_{j+1} - r_j = z_j T^S$, and

$$V(t) = \begin{cases} 0 & if \quad t \bmod T^S \neq 0 \\ V'(\frac{t}{T^S}) & \text{otherwise.} \end{cases} \qquad (2)$$

with $V'(x) = P\{r_{j+1} - r_j = xT^S\} = P\{z_j = x\}$.

Then, the system can be modeled with a queue in which:

1. each T^S units of time, Q^S units of time can be served;
2. a job arrival corresponds to a request of c_j units of time entering the queue;
3. when a job arrives, the next request of c_{j+1} units will arrive after $r_{j+1} - r_j = z_j T^S$ units of time.

Using this model, the scheduling system evolution is described by random process v_j defined as follows:

$$\begin{aligned} v_1 &= c_1 \\ v_j &= max\{0, v_{j-1} - z_j Q^S\} + c_j \end{aligned}$$

where v_j indicates the length of the queue (in time units) immediately after job J_j arrival.

It can be shown [2] that job J_j will finish before time

$$r_j + \left\lceil \frac{v_j}{Q^S} \right\rceil T^S.$$

hence, the probability that the queue length is v_j immediately after a job arrival is a lower bound to the probability that the job finishes before the probabilistic deadline

$$\delta = \left\lceil \frac{v_j}{Q^S} \right\rceil T^S.$$

Let $\pi_k^{(j)} = P\{v_j = k\}$ be the state probability of process v_j, let $U(h) = P\{c_j = h\}$ be the probability that an arriving job requires h units of computation time, and let $V'(x) = P\{r_j - r_{j-1} = xT^S\}$ be the probability that the interarrival time between two consecutive jobs is xT^S. Since c_j and $r_{j+1} - r_j$ are time invariant, $U(h)$ and $V'(x)$ do not depend on j. Under these assumptions, we can compute $\pi_k^{(j)}$ as follows:

$$\pi_k^{(j)} = P\{v_j = k\} =$$

$$= P\{max\{0, v_{j-1} - z_j Q^S\} + c_j = k\} =$$

$$= \sum_{h=-\infty}^{\infty} P\{max\{0, v_{j-1} - z_j Q^S\} + c_j = k \wedge v_{j-1} = h\} =$$

$$= \sum_{x=-\infty}^{\infty} \sum_{h=-\infty}^{\infty} P\{max\{0, v_{j-1} - z_j Q^S\} + c_j = k$$

$$\wedge v_{j-1} = h \wedge z_j = x\}.$$

Being v_j and z_j greater than 0 by definition, the sums can be computed for h and x going from 0 to infinity:

$$\pi_k^{(j)} = \sum_{x=0}^{\infty} \sum_{h=0}^{\infty}$$

$$P\{max\{0, h - xQ^S\} + c_j = k\} P\{v_{j-1} = h\} P\{z_j = x\}$$

$$= \sum_{h=0}^{\infty} \sum_{x=0}^{\infty} P\{max\{0, h - xQ^S\} + c_j = k\} V'(x) \pi_h^{(j-1)} =$$

$$= \sum_{h=0}^{\infty} \sum_{x=0}^{\infty} P\{c_j = k - max\{0, h - xQ^S\}\} V'(x) \pi_h^{(j-1)} =$$

$$= \sum_{h=0}^{\infty} \sum_{x=0}^{\infty} U(k - max\{0, h - xQ^S\}) V'(x) \pi_h^{(j-1)}$$

Hence,

$$\pi_h^{(j)} = \sum_{h=0}^{\infty} m_{h,k} \pi_h^{(j-1)}$$

with

$$m_{h,k} = \sum_{x=0}^{\infty} U(k - max\{0, h - xQ^S\}) V'(x). \qquad (3)$$

Considering $m_{h,k}$ as an element of a matrix M, $\pi_k^{(i)}$ can be computed by solving the equation

$$\Pi^{(j)} = M \Pi^{(j-1)} \qquad (4)$$

where

$$\Pi^{(j)} = \begin{pmatrix} \pi_0^{(j)} \\ \pi_1^{(j)} \\ \pi_2^{(j)} \\ \pi_3^{(j)} \\ . \\ . \\ . \end{pmatrix}.$$

3.1. Stability Considerations

For a generic queue, it is known that the queue is stable (i.e., the number of elements in the queue do not diverge to infinity) if

$$\rho = \frac{mean\ interarrival\ time}{mean\ service\ time} < 1.$$

Hence, the stability can be achieved under the condition

$$\frac{E[C]}{E[T]} < \frac{Q^S}{T^S}$$

where $E[C]$ is the execution time expectation and $E[T]$ is the interarrival time expectation.

If this condition is not satisfied the difference $f_j - r_j$ between the finishing time f_j and arrival the time r_j of each job J_j will increase indefinitely diverging to infinity as j increases:

$$\lim_{j\to\infty} f_j - r_j = \infty.$$

This means that, for preserving the schedulability of the other tasks, τ will slow down in an unpredictable manner.

If a queue is stable, a stationary solution of the Markov chain describing the queue can be found; that is, there exists a finite solution Π such that $\Pi = \lim_{j\to\infty} \Pi^{(j)}$. Since $\Pi^{(j)} = M\Pi^{(j-1)}$, we can compute Π as follows:

$$\Pi = \lim_{j\to\infty} \Pi^{(j)} =$$

$$= \lim_{j\to\infty} M\Pi^{(j-1)} =$$

$$= M \lim_{j\to\infty} \Pi^{(j-1)} = M\Pi.$$

Hence, Π can be computed by solving the eigenvector problem

$$\Pi = M\Pi.$$

This solution can be approximated by truncating the infinite dimension matrix M to an NxN matrix M' and solving the eigenvector problem $\Pi' = M'\Pi'$ with some numerical calculus technique.

4. Relaxing the hypothesis on interarrival times

In our previous analysis, we assumed that task interarrival times are multiple of an integer value T^S so that Equation (2) is verified. This assumption results to be very useful in order to simplify the queue analysis, but can be unrealistic in some practical situations.

In this section, we will show how to relax the assumption on the interarrival times without compromising the analysis based on it. When Equation (2) is not respected, we can introduce a new distribution $\tilde{V}(t)$ which approximates $V(t)$ for enabling the analysis developed in Section 3. In this way, it is possible to perform the analysis based on the approximate PDF $\tilde{V}(t)$ instead of the actual PDF $V(t)$. In order this approximation to be correct, $\tilde{V}(t)$ must:

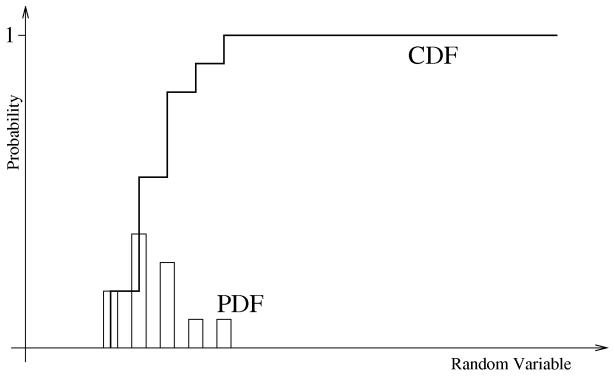

Figure 1. CDF vs PDF.

- be conservative (pessimistic);
- verify Equation (2).

By "being conservative", we mean that if a probabilistic deadline can be guaranteed using $\tilde{V}(t)$, it is guaranteed also according to the real distribution $V(t)$. Since the opposite is not true, this approach is pessimistic.

The new PDF $\tilde{V}(t)$ is conservative if

$$\forall k, \sum_{i=0}^{k} \tilde{V}(i) \geq \sum_{i=0}^{k} V(i), \tag{5}$$

while the second requirement states that

$$\tilde{V}(t) = \begin{cases} 0 & if \quad t \bmod T^S \neq 0 \\ V'(\frac{t}{T^S}) & \text{otherwise.} \end{cases}$$

Equation (5) states that the approximated interarrival times Cumulative Distribution Function (CDF) $\tilde{W}(t)$ computed from $\tilde{V}(t)$ must be greater than or equal to the interarrival times CDF computed from $V(t)$. We recall that the CDF of a stochastic variable expresses the probability that the variable is less than or equal to a given value. The CDF W of a stochastic variable t can be computed as $W(t) = \sum_{i=0}^{t} V(t)$, where $V(t)$ is the PDF of t, as shown in Figure 1. In practice, the intuitive interpretation of Equation 5 is that a $\tilde{V}(t)$ is conservative if the probability that the interarrival time is smaller than t according to $\tilde{V}(t)$ is bigger than according to $V(t)$. Figure 2 explains this concept.

Given a generic interarrival times PDF $V(t)$, it is possible to generate a conservative approximation $\tilde{V}(t)$ if $\exists k$:

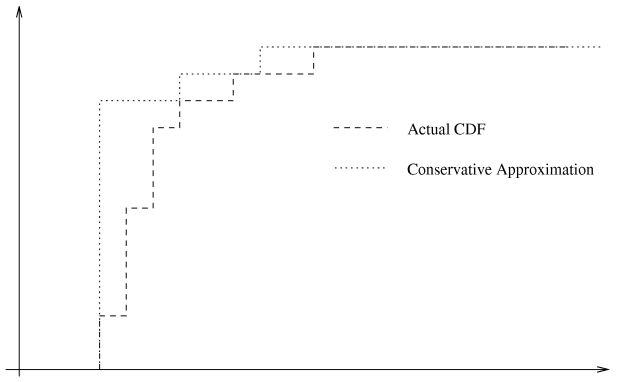

Figure 2. Conservative approximation of a CDF.

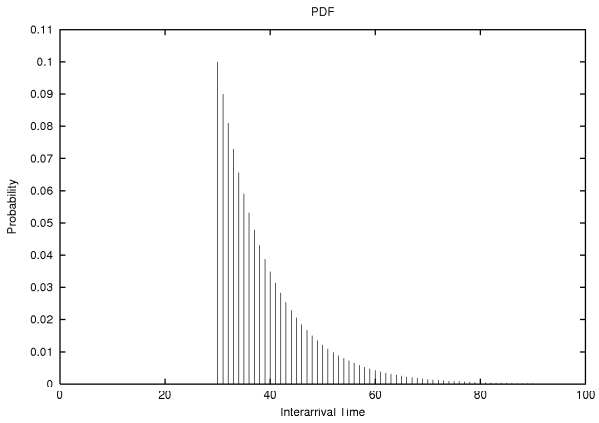

Figure 3. Interarrival times PDF.

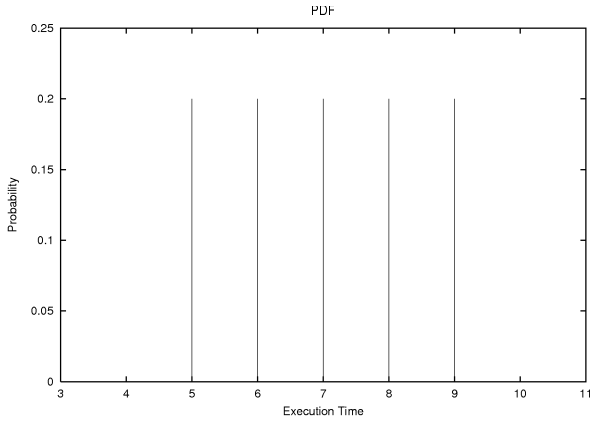

Figure 4. Execution times PDF.

$t < k \Rightarrow V(t) = 0$. In this case, it is possible to set $T^S < k$ and to compute

$$\tilde{V}(t) = \begin{cases} 0 & if \quad t \bmod T^S \neq 0 \\ \sum_{i=t-T^S+1}^{t} V(t) & \text{otherwise.} \end{cases} \quad (6)$$

It can be easily verified that computing $\tilde{V}(t)$ in this way, it will have both the required properties. However, every conservative approximation $\tilde{V}(t)$ respecting Equation (5) can be used: an extreme case is using

$$\tilde{V_0}(t) = \begin{cases} 1 & if \quad t = T^S \\ 0 & \text{otherwise.} \end{cases}$$

This very pessimistic approximation corresponds to the worst-case sporadic task, that is a periodic task with period equal to the MIT. In this case, we have

$$V'(x) = \begin{cases} 1 & if \quad x = 1 \\ 0 & \text{otherwise} \end{cases}$$

and Equation (3) becomes

$$m_{h,k} = U(k - max\{0, h - Q^S\})$$

that is coherent with the results in [2].

The $\exists k : t < k \Rightarrow V(t) = 0$ assumption is realistic (an interarrival time must have a lower bound) and does not impose serious limits to the applicability of the analysis. However, in some occasions the lower bound can be too small, resulting in a small T^S value that unnecessarily increases the number of context switches; in some other cases, a continuous distribution can be used to approximate $V(t)$, making difficult to compute the lower bound.

In these cases, some approximation can be introduced by truncating the interarrival times PDF. In practice, this can be done by considering $V(t) = 0$ if $t < t_0$, with $V(t_0) << 1$; in this way, we can assign $T^S \leq t_0$.

5. An Example

The usefulness and the power of the proposed analysis can be better understood through a simple example: consider task τ with interarrival times distributed as in Figure 3 and computation times uniformly distributed between 5 and 9, as shown in Figure 4.

As a first step, we have to generate an approximation $\tilde{V}(t)$ of the interarrival time PDF. Since the minimum interarrival time is known and is equal to $MIT = 30$, it is possible to use the methodology explained in Section 4 for approximating $V(t)$. Applying Equation (6) with $Q^S = \{30, 20, 10, 5\}$, we obtain the approximated PDFs $\tilde{V}(T)$ shown in Figure 5. Table 1 shows the mean interarrival time computed according to the $V(t)$ PDF and to the $\tilde{V}(t)$ approximations, and the minimum Q^S parameters that can be used for obtaining a stable queue.

It is possible to see how, in general, bigger periods T^S introduce more pessimistic approximations (but on the other hand, reduce the number of context switches). Another im-

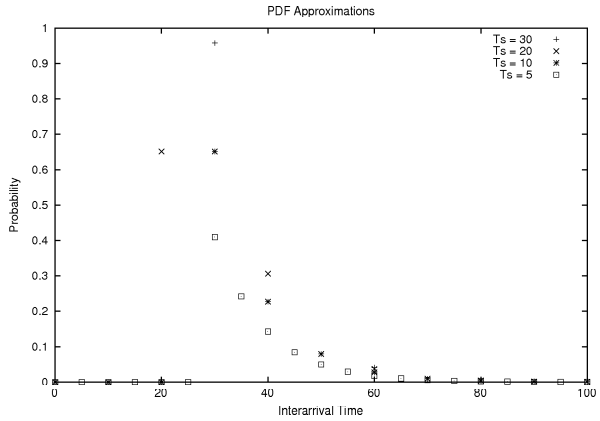

Figure 5. Approximations of the interarrival times distribution for different values of T^S.

PDF	Mean IAT	minimum Q^S
$V(t)$	39	—
$\tilde{V}(t), T^S = 30$	31.327740	7
$\tilde{V}(t), T^S = 20$	27.938438	6
$\tilde{V}(t), T^S = 10$	35.353107	2
$\tilde{V}(t), T^S = 5$	37.209408	1

Table 1. Mean interarrival times computed according to different approximations.

portant thing to note is that $T^S = 20$ results to be the most pessimistic between the considered values (in facts, it is the one that gives the smallest mean interarrival time): this is due to the fact that the minimum interarrival time (that is 30) is not a multiple of it.

For each of the approximations, the probabilistic deadline PDF can be computed solving the eigenvector problem presented in Section 3. Some of the obtained results are shown in Figure 9 (for $T^S = 5$), Figure 8 ($T^S = 10$), Figure 7 ($T^S = 20$), and Figure 6 ($T^S = 30$). Analyzing those figures and comparing the plots with the system requirements, it is possible to choose the correct (T^S, Q^S) pair for task τ.

6. Conclusions

In this paper, we showed how to perform QoS guarantee through *probabilistic deadlines* when a resource reservation mechanism is used for soft tasks scheduling. The presented methodology enables a system designer to perform a new kind of analysis of real-time systems, permitting to guarantee a deadline completion probability for each different task (hard real-time tasks can be considered tasks with a deadline

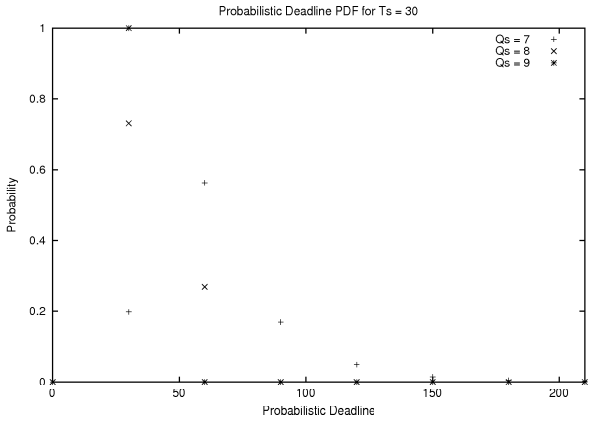

Figure 6. Probabilistic deadlines for $T^S = 30$.

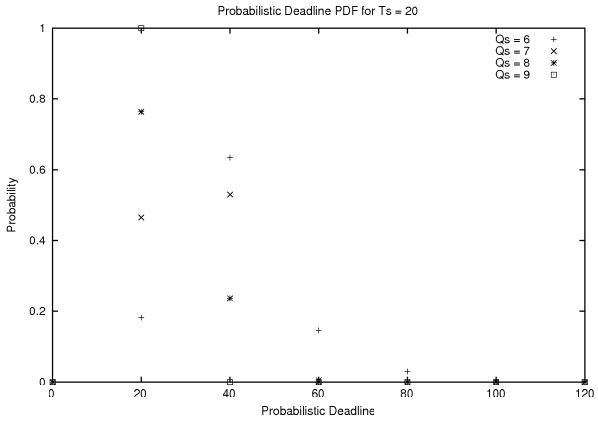

Figure 7. Probabilistic deadlines for $T^S = 20$.

completion probability equal to 1). The only requirements needed for applying the proposed approach are:

1. a reservation based scheduler (no matter of the specific scheduling algorithm that is used);
2. the knowledge of the execution times and interarrival times PDFs (an approximate PDF can also be used).

As a future work we plan to investigate the impact that an approximation in the PDF can introduce on the overall guarantee. Another interesting extension to this work can be to use some closed-form function to approximate the PDF, in order to develop a closed form solution instead of using numerical techniques.

We also intend to simplify the guarantee mechanism in order to make it usable on-line for runtime resource management (on-line QoS guarantee). In this case, we need a method for automatically deriving the interarrival and execution times PDFs at runtime, using system measurements as a feedback.

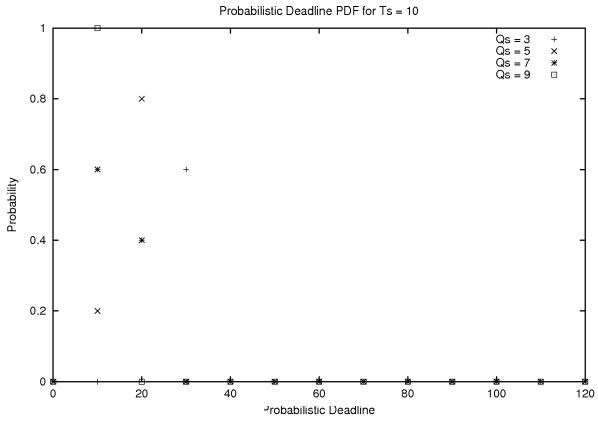

Figure 8. Probabilistic deadlines for $T^S = 10$.

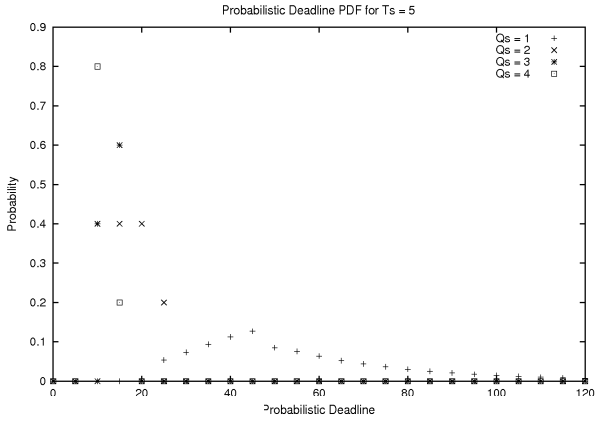

Figure 9. Probabilistic deadlines for $T^S = 5$.

References

[1] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the IEEE Real Time Systems Symposium*, Madrid, Spain, December 1998.

[2] L. Abeni and G. Buttazzo. Qos guarantee using probabilistic dealines. In *Proceedings of the IEEE Euromicro Conference on Real-Time*, York, England, June 1998.

[3] A. K. Atlas and A. Bestavros. Statistical rate monotonic scheduling. In *IEEE Real Time System Symposium*, Madrid, Spain, December 1998.

[4] Z. Deng and J. W. S. Liu. Scheduling real-time applications in open envirovment. In *IEEE Real-Time Systems Symposium*, December 1997.

[5] Z. Deng, J. W. S. Liu, and J. Sun. A scheme for scheduling hard real-time applications in open system environment. In *Ninth Euromicro Workshop on Real-Time Systems*, 1997.

[6] M. K. Gardner and J. W. Liu. Analysing stochastic fixed-priority real-time systems. In *Joint European Conferences on Theory and Practice of Software*, March 1999.

[7] M. B. Jones, P. J. Leach, R. P. Draves, and J. S. Barrera. Modular real-time resource management in the rialto operating system. In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, WA, May 1995.

[8] D.-I. Kang, R. Gerber, and M. Sakena. Performance-based design of distributed real-time systems. In *IEEE Real-Time Technology and Applications Symposium*, pages 2–13, June 1997.

[9] J. P. Lehoczky. Real-time queueing theory. In *Proceedings of the IEEE Real Time Systems Symposium*, December 1996.

[10] J. P. Lehoczky. Real-time queueing network theory. In *Proceedings of the IEEE Real Time Systems Symposium*, December 1997.

[11] J. P. Lehoczky. Using real-time queueing theory to control lateness in real-time system. *Performance Evaluation Review*, 1(25):158–168, 1997.

[12] I. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal on Selected Area on Communications*, June 1997.

[13] J. Leung and J. W. Whitehead. On the complexity of fixed priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4), 1982.

[14] C. L. Liu and J. Layland. Scheduling alghorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.

[15] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *In Proceedings of the IEEE International Conference on Multimedia Computing and Systems Computing System*, May 1994.

[16] L. Palopoli, L. Abeni, F. Conticelli, M. D. Natale, and G. Buttazzo. Real-time control system analysis: An integrated approach. In *Proceedings of the Real Time Linux Workshop*, Orlando, Florida, December 2000.

[17] R. Rajkumar, L. Abeni, D. D. Niz, S. Gosh, A. Miyoshi, and S. Saewong. Recent developments with linux/RK. In *Proceedings of the Real Time Linux Workshop*, Orlando, Florida, December 2000.

[18] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[19] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J. W.-S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Real-Time Technology and Applications Symposium*, pages 164–173, Chicago,Illinois, January 1995.