# A Stochastic Approach for Fine Grain QoS Control

Jacques Combaz, Loïc Strus
Verimag, Centre Equation - 2 avenue de Vignate F38610 Gières, France

## Abstract

*We present a method for fine grain QoS control of multimedia applications. This method takes as input an application software composed of actions parameterized by quality levels. Our method allows the construction of a Quality Manager which computes adequate action quality levels, so as to meet QoS requirements (action deadlines are met and quality levels are maximal) for a given platform.*

*We have developed a stochastic approach based on probability distribution functions for the execution time of the actions. Our method is parameterized according to the importance attributed to deadline misses. This means that the user is given the possibility to express how hard the real-time constraints are. Besides, given a value of the parameter, we can compute the expected deadline miss ratio for the controlled application.*

*We present experimental results including the implementation of the method and benchmarks for an MPEG4 video encoder.*

## 1. Introduction

Designing systems meeting both hard and soft real-time requirements is a challenging problem. There exist well-established design methodologies for hard real-time systems, that is, systems that do not violate critical properties such as deadlines. These methodologies are based on worst-case analysis using conservative approximations of the system dynamics and static resource reservation. This implies high predictability but a non optimal use of resources.

In contrast, design methodologies for soft real-time are based on average-case analysis and seek more efficient use of resources (e.g. optimization of speed, jitter, memory, bandwidth, power) without addressing critical behavior issues. They are used for applications where some degradation or even temporal denial of service is tolerated, e.g., multimedia and telecommunications.

These two classes of design methodologies are currently disjoint. Meeting hard real-time properties and making optimal use of available resources seem to be two antagonistic requirements. The existing gap between hard and soft real-time often leads to costly and unreliable solutions. Development of soft real-time approaches that ensure predictability is a key challenge in the design of modern methodologies for real-time embedded systems.

Our method targets multimedia applications. It allows adapting the overall system behavior by adequately setting quality level parameters for its actions. The objective of the quality management policy is to meet QoS requirements including three types of properties: 1) safety (no deadlines are missed); 2) optimality, (maximization of the utilization of available time budget); 3) smoothness of quality levels.

The method takes as input an application software with timing information about its actions. This includes deadlines and (platform-dependent) execution times. It produces a controlled application software meeting the QoS requirements for the target platform. This is obtained by applying to the application software a *Controller* consisting of a *Scheduler* and a *Quality Manager*. Depending on the progress of the computation, the Scheduler chooses the next action to be executed and the Quality Manager computes the associated quality level parameter.

In [5], we explained how to build quality management policies meeting QoS requirements. We also provided low overhead implementations of the controller in [6].

In this paper, we present a *stochastic* approach for computing quality management policies. It uses probability distribution functions representing varying execution times. The proposed quality management policies are parameterized according to the importance attributed to deadline misses. This means that the user is given the possibility to express how hard the real-time constraints are. Besides, given a value of the parameter, we provide tools for computing the expected deadline miss ratio for the controlled application.

We consider the following simplified version of the general problem by assuming that the application software is already scheduled:

- The application software cyclically performs input/output transformations of data streams. It is described as a finite sequence of actions. Its execution during a cycle can be controlled by choosing *quality level parameters*.
- We consider single-thread implementations of the application software on a platform for which it is possible, by

using timing analysis and/or profiling techniques, to compute probability distributions of execution times of actions for different quality levels. Action execution is atomic.

The controlled software can be considered as the composition of the initial application software with a *Quality Manager* (see figure 1). The latter monitors the progress of the computation within a cycle of the application software. At any state of the cycle, it chooses the quality level for the next action to be executed, guided by a *quality management policy*. This is a constraint guaranteeing safety and embodying an optimality criterion. The Quality Manager chooses the maximal quality satisfying this constraint.

Our method significantly differs from existing ones. The main difference is fine granularity of quality management, which allows combination of hard and soft real-time techniques. Most existing techniques are applied at system or task level, focus on average scenario and do not provide predictability. Taking into account worst-case scenario is useful in applications where quality should remain above some minimal level [2], e.g., home TVs. Buttazzo et al.'s elastic tasks model [3], as well as slack scheduling [7], [10] and gain time techniques [1] are based only on worst-case execution times and do not deal with quality smoothness. Lu et al. [11] propose a feedback scheduling based on PID controllers. Steffens et al. [13] minimize deadline misses of an MPEG decoder by applying a Markov decision process and reinforcement learning techniques, combined with structural load analysis. A few papers have studied performance estimation for tasks with varying computation times [9],[8],[12],[14]. Kalavade and Mogh [9] studied the timing performance of networked embedded systems, which may have precedence constraints. The authors proposed an algorithm which aim at finding a probability of each task meeting its timing constraints. Their algorithm is based on a Markovian stochastic processes.

The paper is organized as follows. In section 2 we present the quality management problem. A stochastic quality manager and performance analysis tools are presented in section 3. Section 4 presents experimental results for a non trivial MPEG4 video encoder.

## 2. Quality Management

### 2.1. Definition of the Problem

We provide a formalization of the quality management problem by considering that the application software is already scheduled. It is characterized by an execution sequence $\{ s_{i-1} \xrightarrow{a_i} s_i \}_{1 \leq i \leq n}$, where $S = \{ s_0, \ldots, s_n \}$ is a set of *states* and for all $i$ we have $a_i \in A$ where $A$ is a finite set of *actions*. Actions correspond to blocks of code, their execution is atomic.

Execution times for actions may considerably vary over time as they depend on the contents of data. Furthermore, non predictability of the underlying platform is an additional factor of uncertainty. We consider that they are not known in advance, but are characterized by discrete probability distributions. These can be obtained by profiling techniques. To cope with the inherent uncertainty of execution times, we assume that actions are parameterized by quality levels. This leads to the following model.

**Definition 1.** *A* **parameterized system** $PS$ *is an application software* $(A, S)$ *with*
• *a finite set of integer* quality levels $Q$
• *a set of independent and discrete random variables* $\{ C(a_i, q) \in \mathbb{N} \}_{a_i \in A, q \in Q}$, *representing the* actual *execution times of the actions*
• *for each action* $a_i \in A$ *and quality level* $q \in Q$, *we assume that* $d^q_{a_i} : \mathbb{N} \to [0, 1]$ *is the probability distribution function of* $C(a_i, q)$, *that is,* $d^q_{a_i}(k) = \mathrm{P}\big[C(a_i, q) = k\big]$ *where* $\mathrm{P}\big[C(a_i, q) = k\big]$ *denotes the probability of* $C(a_i, q) = k$, *and it satisfies* $\sum_{k \geq 0} d^q_{a_i}(k) = 1$.

*The execution of a parameterized system is characterized by the family of sequences* $\{ (s_{i-1}, t_{i-1}) \xrightarrow{a_i, q_i} (s_i, t_i) \}_{1 \leq i \leq n, q_i \in Q}$ *such that* $t_0 = 0$ *and* $t_i - t_{i-1} = C(a_i, q_i)$.

Quality Managers are used to restrict the behavior of a parameterized system so as to meet given properties.

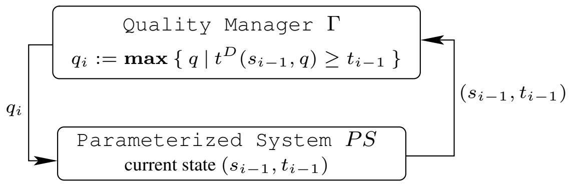
**Definition 2.** *Given a parameterized system* $PS$ *a* **Quality Manager** *is a function* $\Gamma : S \times \mathbb{N} \to Q$ *giving, for a state* $(s_{i-1}, t_{i-1})$ *of* $PS$, *the quality level* $q_i$ *for executing the next action* $a_i$.

$PS||\Gamma$ *denotes a* controlled system *obtained as the composition of the parameterized system* $PS$ *and the Quality Manager* $\Gamma$. *For given values of execution times* $C(a_i, q)$, *it has a single execution sequence* $\{ (s_{i-1}, t_{i-1}) \xrightarrow{a_i, q_i} (s_i, t_i) \}_{1 \leq i \leq n}$ *such that* $q_i = \Gamma(s_{i-1}, t_{i-1})$.

*Given a* deadline $D \in \mathbb{N}$, *we denote by* $\mu(\Gamma)$ *the deadline miss ratio for the Quality Manager* $\Gamma$. *It is defined by the probability for missing the deadline in* $PS||\Gamma$, *that is* $\mu(\Gamma) = \mathrm{P}\big[t_n > D\big]$ *where* $t_n$ *is the completion time of the last action in* $PS||\Gamma$.

The quality management problem for a given parameterized system $PS$ consists in finding a Quality Manager $\Gamma$ meeting the QoS requirements. That is, deadline is met with a minimal probability and the overall quality is maximal. It is formalized as follows.

**Definition 3** (quality management problem)**.** *Given a parameterized system* $PS$, *a deadline* $D$ *and a target deadline miss ratio* $\lambda \in [0, 1]$, *find a Quality Manager* $\Gamma$ *such that:*
• $\Gamma$ *has a maximal deadline miss ratio of* $\lambda$: $\mu(\Gamma) \leq \lambda$.
• *The overall execution time is maximal, that is for any Quality Manager* $\Gamma'$ *with a maximal deadline miss ratio of* $\lambda$ *we have* $t_n \geq t'_n$, *where* $t_n$ *(resp.* $t'_n$*) is the completion time of the last action in* $PS||\Gamma$ *(resp.* $PS||\Gamma'$*).*



Quality Manager $\Gamma$
$q_i := \mathbf{max} \{ q \mid t^D(s_{i-1}, q) \geq t_{i-1} \}$
$q_i$
$(s_{i-1}, t_{i-1})$
Parameterized System $PS$
current state $(s_{i-1}, t_{i-1})$

**Figure 1. Quality Manager**

In [5], we require in addition *smoothness* for the chosen quality levels.

## 2.2. Quality Manager Design

Figure 1 shows interaction between the Quality Manager $\Gamma$, applying a *quality management policy*, and the application software, i.e. the parameterized system $PS$. The Quality Manager observes the current state $(s_{i-1}, t_{i-1})$ of $PS$ and computes the next quality level $q_i$ for the next action $a_i$. The Quality Manager is defined by:

$$\Gamma(s_{i-1}, t_{i-1}) = q_i = \mathbf{max}_{q \in Q}\, t^D(s_{i-1}, q) \geq t_{i-1}.$$

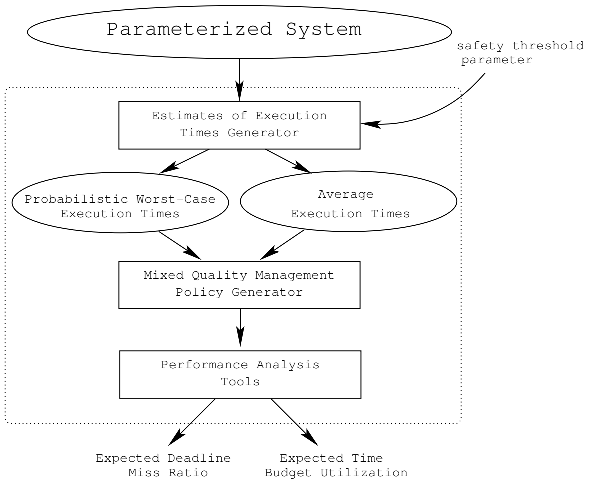
The function $t^D : S \times Q \to \mathbb{N}$ defines the *quality management policy* of the Quality Manager. It gives for a state of the application software $s_{i-1}$ and a quality level $q$ the estimated elapsed time $t^D(s_{i-1}, q)$ if the remaining actions are executed with constant quality $q$. If the inequality $t^D(s_{i-1}, q) \geq t_{i-1}$ is satisfied, then it is possible to complete execution with the quality level $q$. The chosen quality level $q_i$ at state $(s_{i-1}, t_{i-1})$ is maximal amongst the quality levels $q$ meeting the inequality $t^D(s_{i-1}, q) \geq t_{i-1}$. The function $t^D$ is defined by $t^D(s_{i-1}, q) = D - C^D(a_i..a_n, q)$, where $C^D(a_i..a_n, q)$ denotes an estimation of the total execution time for the sequence of actions $a_i, a_{i+1}, \ldots, a_n$.

Choosing an adequate quality management policy, i.e. meeting QoS requirements, is a non trivial problem discussed in [4] and [5]. In [5] we proposed the *mixed* quality management policy based on average and worst-case estimates of execution times. To cope with overestimated worst-case execution times, the next section propose a stochastic adaptation of the mixed quality management policy by using *probabilistic* worst-case execution times.

## 3. Stochastic Quality Manager

The quality management policy proposed in this section is a stochastic adaptation of the mixed quality management policy defined in [5]. It is obtained from the latter by replacing worst-case estimates of execution times with *probabilistic* worst-case estimates. We introduce the *safety threshold* that is a parameter for building these estimates.

As actual execution times may exceed probabilistic worst-case estimates, using stochastic quality management policy may lead to deadline misses. For that reason we provide an algorithm for computing the expected deadline miss ratio of the controlled application.



Parameterized System
safety threshold parameter
Estimates of Execution Times Generator
Probabilistic Worst-Case Execution Times
Average Execution Times
Mixed Quality Management Policy Generator
Performance Analysis Tools
Expected Deadline Miss Ratio
Expected Time Budget Utilization

**Figure 2. Quality management tool chain**

### 3.1. Estimates of Execution Times

Given an action $a_i$ and a quality level $q$, we define the average execution time as the mean value $\mathrm{E}(d^q_{a_i})$ of the corresponding probability distribution function $d^q_{a_i}$.

**Definition 4.** *Given a parameterized system $PS$, we define the* **average** *execution time function $C^{av} : A \times Q \to \mathbb{N}$ by* $C^{av}(a_i, q) = \mathrm{E}(d^q_{a_i}) = \sum_{k \geq 0} k d^q_{a_i}(k)$.

The computation of probabilistic worst-case execution times is parameterized by the *safety threshold* $\tau \in [0, 1]$. These worst-case estimates are computed so that the probability of exceeding them is lower than $\tau$.

**Definition 5.** *Given a parameterized system $PS$ and a safety threshold $\tau$, we define the* **probabilistic worst-case** *execution time function $C^{wc}_\tau : A \times Q \to \mathbb{N}$ such that* $\mathrm{P}\big[C(a_i, q) > C^{wc}_\tau(a_i, q)\big] \leq \tau$*, that is:*

$$C^{wc}_\tau(a_i, q) = \mathbf{min}_{k \geq 0} \sum_{l > k} d^q_{a_i}(l) \leq \tau.$$

### 3.2. Stochastic Quality Management Policy

The stochastic quality management policy $C^{mx}_\tau$ considered in this paper is obtained from mixed quality management policy [5] by replacing worst-case execution times with probabilistic worst-case execution times, that is, $C^{mx}_\tau = C^{av} + \delta^{max}_\tau$, where $\delta^{max}_\tau(a_i..a_n) = \mathbf{max}\ \{ 0 \} \cup \{ C^{sf}_\tau(a_j..a_n, q) - C^{av}(a_j..a_n, q) \mid i \leq j \leq n \}$ and $C^{sf}_\tau(a_j..a_n, q) = C^{wc}_\tau(a_i, q) + C^{wc}_\tau(a_{i+1}, q_{min}) + .. + C^{wc}_\tau(a_n, q_{min})$.

As actual execution times may exceed probabilistic worst-case estimates, using stochastic quality management policy may lead to deadline misses. Setting $\tau$ to 0 implies that the probability for exceeding probabilistic worst-case execution times $C^{wc}_0$ is null. In other words, $C^{wc}_0$ are exact worst-case execution times (i.e. actual execution times never exceed them), with the assumption that probability distribution functions $d^q_{a_i}$ are an exact model of execution times of actions. For $\tau = 0$ the stochastic quality management policy is equivalent to the standard mixed quality management policy. It ensures that no deadline is missed.

On the contrary, the probabilistic worst-case execution time function $C^{wc}_\tau$ tends to 0 as $\tau$ tends to 1. As a result, using $\tau = 1$ (i.e. $C^{wc}_1 = 0$) leads to stochastic quality management policies only based on average execution times, without addressing deadline miss ratio minimization.

The stochastic quality management policy provides flexibility in the combination of the average and worst-case behavior. The safety threshold will determine how much the stochastic quality management policy takes the worst-case scenario into account. It ranges from no taking into account the worst-case — which corresponds to pure soft real-time ($\tau = 1$) — to maximal taking into account — which corresponds to hard real-time, that is, no deadline is missed ($\tau = 0$). In some sense, the parameter $\tau$ is used to express how hard the deadline $D$ is. Our method helps designers to achieve good compromises between deadline miss ratio and resource utilization.

### 3.3. Performance Analysis

In this section we provide a method allowing prediction of the behavior of the controlled application. We develop an algorithm that computes probability distribution functions for the completion time of the actions.

The completion time $t_i$ of an action $a_i$ in the controlled system $PS||\Gamma$ is defined recursively by $t_0 = 0$ and $t_i = t_{i-1} + C(a_i, q_i)$, where $q_i$ is only determined by the value of $t_{i-1}$, that is, $q_i = \Gamma(s_{i-1}, t_{i-1})$. We denote by $d_i : \mathbb{N} \to [0, 1]$ the probability distribution function of $t_i$. It is defined by $d_i(k) = \mathrm{P}\big[t_i = k\big]$ where $\mathrm{P}\big[t_i = k\big]$ is the probability of $t_i = k$.

Let $k$ be an integer. As $\{ \ t_i = k \wedge t_{i-1} = l \ \}_{l\in\mathbb{N}}$ are mutually exclusive events, we obtain:

$$\mathrm{P}\big[t_i = k\big] = \sum_{l\geq 0} \mathrm{P}\big[t_i = k \wedge t_{i-1} = l\big]. \qquad (1)$$

As $t_i = t_{i-1} + C(a_i, q_i)$, we have:

$$\mathrm{P}\big[t_i = k \wedge t_{i-1} = l\big] = \mathrm{P}\big[t_{i-1} = l \wedge C(a_i, q_i) = k - l\big].$$

Since $t_i = C(a_1, q_1) + \ldots + C(a_{i-1}, q_{i-1})$ and execution times of actions are independent, we obtain:

$$\mathrm{P}\big[t_i = k \wedge t_{i-1} = l\big] = \mathrm{P}\big[t_{i-1} = l\big]\mathrm{P}\big[C(a_i, q_i) = k - l\big].$$

The above equation and (1) leads to the following relationship between distribution functions:

$$d_i(k) = \sum_{l\geq 0} d_{i-1}(l) d^{q_i}_{a_i}(k - l). \qquad (2)$$

Algorithm 1 given below is based this relationship. We assume that uninitialized values are equal to 0. The order of computation in the algorithm slightly differs from the one of (2). Given integers $l$ and $t$, we increment the value of $d_i(l + t)$ by $d_{i-1}(l)d^{q_i}_{a_i}(t)$, which is equivalent to (2).

It can be shown that the proposed algorithm is polynomial in the number of actions and in the length of the distributions of execution times. However, it can easily be accelerated by using Fast Fourier Transform for computing convolutions.

```
parametric_convolution($\{ d^{q_i}_{a_i} \}_{1\leq i\leq n}$,$\Gamma$)
$i = 1$
$d_0(0) = 1$
for $i \leftarrow 1$ to $n$ do
    foreach $l$ such that $d_{i-1}(l) \neq 0$ do
        $q_i = \Gamma(s_{i-1}, l)$
        foreach $t$ such that $d^{q_i}_{a_i}(t) \neq 0$ do
            $d_i(l + t) \leftarrow d_i(l + t) + d_{i-1}(l) * d^{q_i}_{a_i}(t)$
```

**Algorithm 1**: Parametric convolution

Performance evaluation of the Quality Manager is done by measuring deadline miss ratio, that is, probability of deadline misses, as well as expected time budget utilization, that is, the expected overall execution time $t_n$. The algorithm proposed above allows computing the distribution $d_n$ of $t_n$, depending on the safety threshold parameter $\tau$.

**Expected Deadline Miss Ratio.** The deadline miss ratio is defined by $\mathrm{P}\big[t_n > D\big]$ (see definition 2). Since $d_n$ is the probability distribution function of the random variable $t_n$, the expected deadline miss ratio for the controlled application is defined by $\sum_{k>D} d_n(k)$.

**Expected Time Budget Utilization.** The expected time budget utilization is computed as the mean value of the distribution $d_n$, that is, $\mathrm{E}(t_n) = \sum_{k\geq 0} k d_n(k)$.

## 4. Experimental Results

This section provides experimental results for an MPEG4 video encoder. They confirm the interest of theoretical sections.

### 4.1. Experimental Framework

We applied our results to an MPEG4 video encoder written in C (more than 10,000 lines of code). The encoder cyclically treats frames. Each frame is split into $N$ macroblocks of $16 \times 16$ pixels. In the following, we consider frames of $320 \times 144$ pixels ($N = 180$). Encoding each macroblock is done by 5 actions: `Grab_Macroblock`, `Motion_Estimation`, `DCTQuantIQuantIDCT`, `IntraPredictionCoding` and `Reconstruction`. The action `Motion_Estimation` is parameterized by a quality level $q \in Q = \{ \ 0, \ldots, 8 \ \}$.

The other actions do not depend on the quality level $q$. As a result, the action `Motion_Estimation` is said *controllable* while the others are said *uncontrollable*. The probability distribution functions $d^q_{a_i}$ have been computed by using profiling techniques. Figure 3 gives examples of distributions for `Motion_Estimation`.

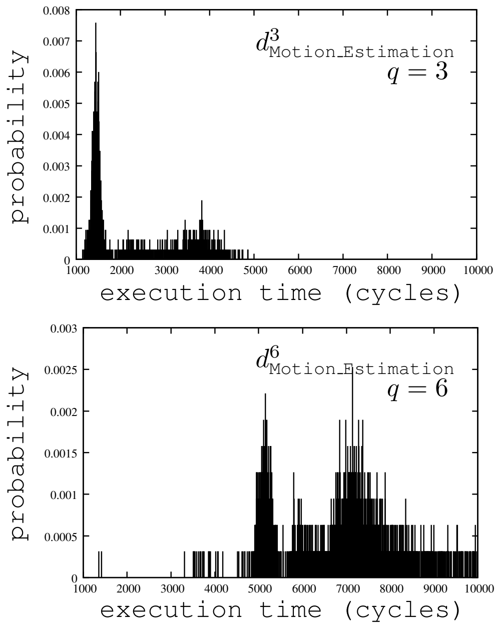


**Figure 3. Distributions for** `Motion_Estimation`

The considered application corresponds to a videophone application. It captures a sequence of frames with a camera, transmits the sequence, and then displays the frames on a screen. The target platform is an STm8010 board from STMicroelectronics. A register that counts the number of processor cycles elapsed provides a real-time clock with minimal access overhead.

We developed a prototype tool that allows the generation of the controlled application software (see figure 2). The inputs of the tool are an application software, probability distributions, the deadline, and the safety threshold parameter $\tau$. From these inputs, the tool computes C code corresponding to the application software, and tables containing pre-computed values used by the Quality Manager. We have considered two different schedules (schedules #1 and #2) composed of the same actions, and a deadline $D = 100\,ms$ (i.e. 10 frame/s).

## 4.2. Performance Analysis

We have computed the expected deadline miss ratio $\mu(\Gamma^{mx}_\tau)$ and the expected time budget utilization $E(t_n)$ for different values of the safety threshold parameter $\tau$ (see Figure 4). The expected time budget utilization $E(t_n)$ are given in percentage of the deadline $D$. As explained in section 3.3, the computation of $\mu(\Gamma^{mx}_\tau)$ and $E(t_n)$ requires the computation of the probability distribution $d_n$ of the completion time $t_n$ of the last action $a_n$.

The deadline miss ratio and the time budget utilization increase as the safety threshold $\tau$ increases, which confirms theoretical definitions of section 3.2. For values of $\tau$ sufficiently high ($\tau > 0.4$), the deadline miss ratio as well as the time budget utilization are constant. This is due to the fact that, for these values of $\tau$, the stochastic quality management policy is equal to the average policy (i.e. $C^{mx}_\tau = C^{av}$). The video encoder behaves then as a pure soft real-time application.

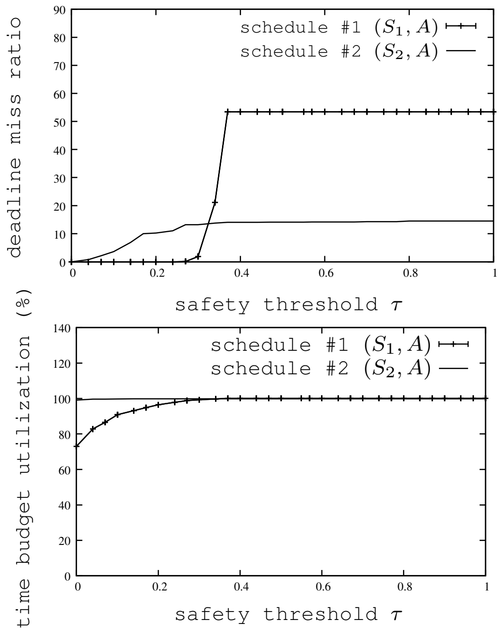


**Figure 4. Expected time budget utilization and deadline miss ratio**

The difference between the results obtained with schedules #1 and #2 comes from the position of controllable actions in the schedule. Controllable actions are scattered all along schedule #2, whereas they are put together at the beginning of schedule #1. Consequently, the Quality Manager keeps control on the execution times of actions during the execution of schedule #2. On the contrary, once all controllable actions have been executed in schedule #1, the Quality Manager has no control anymore on the (uncertain) execution times of the remaining (uncontrollable) actions.

Uncontrollability combined with unpredictability leads to poor performances when using schedule #1. Figure 4 shows an important amount of time budget lost by the application for $\tau < 0.2$. In addition, the deadline miss ratio reaches $50\%$ for $\tau > 0.4$ (see Figure 4). For such an application, our approach can be useful for choosing adequate compromises between the deadline miss ratio and the time budget utilization, depending on user requirements.

Results obtained when running the application with

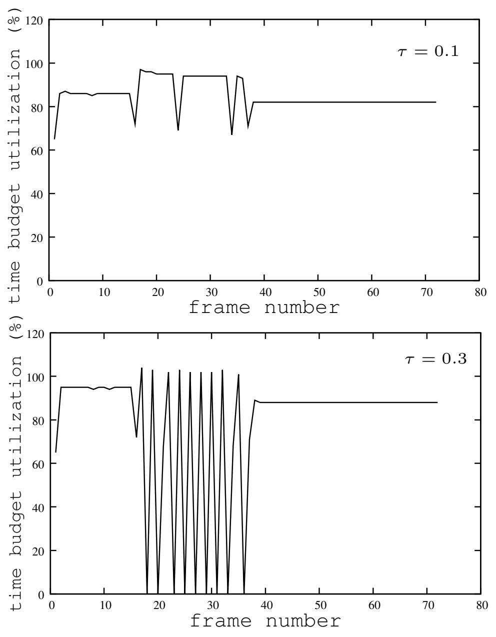


**Figure 5. Actual time budget utilization for schedule #1**

schedule #1 on the target platform confirm expected values of the deadline miss ratio and the time budget utilization computed by our performance analysis tool. In Figure 5, bursts of jumps correspond to frame skips occurring when deadlines are missed. Notice that no deadline is missed for an application running with $\tau = 0.1$. The amount of deadline misses significantly increases for $\tau = 0.3$. The time budget utilization is, in average, $85\%$ for $\tau = 0.1$ and $95\%$ for $\tau = 0.3$.

Results obtained for schedule #2 confirm that (standard) mixed quality management policy is still useful for some applications. In fact, schedule #2 allow running the application with $\tau = 0$ without significant unused time budget.

## 5. Conclusion

Our approach reduces the impact of the worst-case estimates of system behavior by using quality management techniques and probabilistic worst-case estimations of execution times. A parameter is used to determine how much the stochastic quality management policy takes the worst-case scenario into account, ranging from no taking into account the worst-case — which corresponds to pure soft real-time — to maximal taking it into account — which corresponds to hard real-time, that is, no deadline is missed. In some sense, the parameter is used to express how hard the deadlines are.

Our method can considerably help engineers to design real-time applications that are fitted to QoS requirements. In addition, it can improve code reuse as well as reliability.

Choices of the Quality Manager would be more accurate if dependencies between the execution times were considered. We plan to work on models and quality management policies that include these dependencies.

## References

[1] N. C. Audsley, R. I. Davis, and A. Burns. Mechanisms for enhancing the flexibility and utility of hard real-time systems. In *Real-Time Systems Symposium*, pages 12–21. IEEE, 1994.

[2] R. J. Bril, M. Gabrani, C. Hentschel, G. C. van Loo, and E. F. M. Steffens. QoS for consumer terminals and its support for product families. In *Proceedings of the International Conference on Media Futures*, 2001.

[3] G. C. Buttazzo, G. Lipari, and L. Abeni. Elastic task model for adaptive rate control. In *RTSS*, pages 286–295, 1998.

[4] J. Combaz, J. Fernandez, T. Lepley, and J. Sifakis. Fine grain QoS control for multimedia application software. In *Design, Automation and Test in Europe (DATE'05) Volume 2*, pages 1038–1043, 2005.

[5] J. Combaz, J.-C. Fernandez, T. Lepley, and J. Sifakis. QoS Control for Optimality and Safety. In *Proceedings of the 5th Conference on Embedded Software*, September 2005.

[6] J. Combaz, J.-C. Fernandez, J. Sifakis, and L. Strus. Using speed diagrams for symbolic quality management. In *IPDPS*, pages 1–8. IEEE, 2007.

[7] R. I. Davis, K. W. Tindell, and A. Burns. Scheduling slack time in fixed priority preemptive systems. In *Proceeding of the IEEE Real-Time Systems Symposium*, pages 222–231, 1993.

[8] X. S. Hu, T. Zhou, and E. H.-M. Sha. Estimating probabilistic timing performance for real-time embedded systems. *IEEE Trans. Very Large Scale Integr. Syst.*, 9(6):833–844, 2001.

[9] A. Kalavade and P. Moghé. A tool for performance estimation of networked embedded end-systems. In *DAC '98: Proceedings of the 35th annual conference on Design automation*, pages 257–262, New York, NY, USA, 1998. ACM.

[10] J. Lehoczky and S.Thuel. Algorithms for scheduling hard aperiodic tasks in fixed-priority systems using slack stealing. In *Proceedings of the IEEE Real-Time System Symposium*, 1994.

[11] C. Lu, J. Stankovic, G. Tao, and S. Son. Feedback control real-time scheduling: Framework, modeling and algorithm. *special issue of RT Systems Journal on Control-Theoric Approach To Real-TIme Computing*, 23(1/2):85–88, 2002.

[12] S. Manolache, P. Eles, and Z. Peng. Schedulability analysis of applications with stochastic task execution times. *Trans. on Embedded Computing Sys.*, 3(4):706–735, 2004.

[13] C. C. Wüst, L. Steffens, R. J. Bril, and W. F. Verhaegh. QoS control strategies for high-quality video processing. In *Euromicro Conference on Real-Time Systems*, pages 3–12. IEEE, 2004.

[14] T. Zhou, X. S. Hu, and E. H.-M. Sha. A probabilistic performance metric for real-time system design. In *CODES '99: Proceedings of the seventh international workshop on Hardware/software codesign*, pages 90–94, New York, NY, USA, 1999. ACM.