QoS Control for Optimality and Safety

Jacques Combaz[1,2], Jean-Claude Fernandez[1], Thierry Lepley[2], Joseph Sifakis[1]
[1]Verimag, Centre Equation - 2 avenue de Vignate F38610 Gières, France
[2]STMicroelectronics Central R&D 850, rue Jean Monnet 38921 Crolles Cedex, France

ABSTRACT

We propose a method for fine grain QoS control of real-time applications. The method allows adapting the overall system behavior by adequately setting the quality level parameters of its actions. The objective of the control policy is to meet QoS requirements including three types of properties: 1) safety that is, no deadline is missed; 2) optimality that is, maximization of the available time budget; 3) smoothness of quality levels. The method takes as input a model of the application software, QoS requirements and platform-dependent timing information, and produces a controlled application software meeting the QoS requirements on the target platform. This paper provides a complete formalization of the quality control problem. It proposes a new control management policy ensuring safety, near-optimality and smoothness. It also describes a prototype tool implementing the quality control algorithm and experimental results about its application to a video encoder.

Categories and Subject Descriptors

D.1.1 [**Programming Techniques**]: Applicative (functional) Programming; D.2.3 [**Software Engineering**]: Coding Tools and Techniques

General Terms

Algorithms, Performance, Reliability

1. INTRODUCTION

There exist currently two diverging approaches in systems engineering.

• *Critical systems engineering* based on worst-case analysis, using conservative approximations of the system dynamics and static resource reservation. This approach is applied whenever a system's correctness means no violation of critical conditions such as missing a deadline or reaching a dangerous state.

• *Best effort engineering* based on average-case analysis, seeking efficient use of resources without addressing critical behavior issues, e.g., optimization of speed, jitter, memory, bandwidth, power. This approach is applied to applications where some degradation or even temporal denial of service is tolerated e.g., telecommunications.

These two approaches are currently disjoint. They correspond to different research communities and different practices. They adopt different computing paradigms, use specific execution platforms, middleware and networks. It is often advocated that such a separation is inevitable, especially for embedded systems with uncertain execution and external environments. Meeting critical properties and making optimal use of available resources seem to be two conflicting requirements. To ensure critical properties worst-case estimates must be used and this may lead to inefficient use of resources if they are statically pre-allocated as is the case in the current standard practice. The existing gap between critical and best effort approaches often leads to costly and unreliable solutions.

To bridge the gap between the two approaches, it is essential to develop design techniques for adaptive systems meeting both critical and best effort properties. Such techniques should allow to control the overall system behavior so as to meet critical properties while making the best possible use of resources, in spite of the difference between average and worst-case behavior.

This paper contributes to bridging the gap by proposing a method for QoS control allowing optimal use of computing resources without missing deadlines. The method targets multimedia applications. Usually, to meet given QoS requirements, these applications require a significant amount of experimentation on virtual or real prototypes, involving fine tuning of parameters of their software components. After tuning, the behavior of application software can be modified only by changing user-defined input parameters. Thus, adaptability is coarse grain, because it can be achieved only by modifying global parameters. Furthermore, some delay is necessary for adaptation due to the limited controllability of the application software over the underlying execution system. For these applications, uncertainty about execution times dictates the use of control-based techniques [2],[13],[11].

The proposed method allows adapting the overall system behavior by adequately setting the quality level parameters for its actions. The objective of the control policy is to meet QoS requirements including three types of properties: 1) safety that is, no deadline is missed); 2) optimality that is, maximization of the available time budget); 3) smoothness

EMSOFT'05, September 19–22, 2005, Jersey City, New Jersey, USA.

of quality levels. The method takes as input an application software, QoS requirements and platform-dependent timing information and produces a controlled application software meeting the QoS requirements on the target platform, as follows (see figure 1).

• The initial application software cyclically performs input/output transformations of data streams. It is described by a precedence graph modeling dependency between actions (C-functions), and from which all the possible execution sequences can be extracted. Its execution during a cycle can be controlled by choosing *quality levels*, which are parameters of its actions. We assume that the execution times of actions are increasing with quality.

• We consider single-threaded implementations of the application software on a platform for which it is possible, by using timing analysis and profiling techniques, to compute estimates of worst-case execution times and average execution times of actions for the different quality levels. Action execution is assumed to be atomic. A compiler is used to generate the controlled software from the initial application software, for given QoS requirements and execution times.

The controlled software can be considered as the composition of the initial application software with a *controller* (see figure 3). The controller monitors the progress of the computation within a cycle of the application software. At any state of the cycle, it chooses the next action to be executed and its quality level, guided by a quality management policy. This is a constraint guaranteeing safety and embodying an optimality criterion. The optimality criterion is used to compute "best schedules" for different quality parameter levels. The controller chooses amongst these schedules, a feasible one which maximizes quality.

Our method significantly differs from existing ones for QoS control and adaptive scheduling. The main difference is fine grain control of the execution. Existing control techniques act at task level e.g., at the beginning of a cycle, and their reactivity is slow. They do not require any deep knowledge of the data-flow structure of the application software. Our method consists in controlling execution during a cycle; the controlled software is produced by compilation (automatic code instrumentation).

Another important difference is that fine granularity allows combination of optimality and safety of the produced schedules. Most control techniques are applied at system or task level, focus on optimality criteria and are adequate only for soft real-time. The integration of safety criteria is useful in applications where quality should remain above some minimal level [8],[4], e.g., home TVs, or where hard deadlines must be respected. Buttazzo et al.'s elastic tasks model [5], as well as slack scheduling [7], [10] and gain time techniques [3] are proposed in order to adapt a system to its actual behavior, but they are only based on worst-case execution times and do not deal with quality smoothness. A common and simple way to treat CPU overload is to skip an instance of a task [9]. Lu et al. [11] propose a feedback scheduling based on PID controllers, but deadline misses remain possible. Steffens et al. [13],[12] minimize deadline misses of an MPEG decoder by applying a Markov decision process and reinforcement learning techniques, combined with structural load analysis.

The paper improves and extends results presented in [6] in several directions. It provides a complete formalization of the quality control problem. It proposes a new control management policy ensuring better quality smoothness. Finally, it provides a framework for studying how safety and optimality are related depending on system dynamics.

The paper is organized as follows. Section 2 presents a formalization of the quality control problem. Section 3 presents an abstract and generic control algorithm parameterized by a quality management policy. The choice of such policies for safety and quality smoothness is studied in Section 4. Section 5 deals with optimal use of the computing resources. It is shown that safety and optimality are two conflicting requirements and that utilization may differ from the optimal one by a constant depending on the difference between worst-case and average behavior as well as on the granularity of control. Section 6 reports on experimental results. A prototype tool is presented for generating controlled application software, as well as a non trivial example.

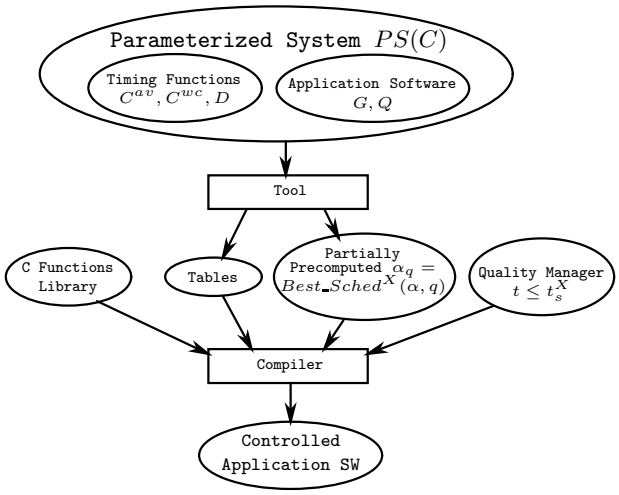

Figure 1: Implementation of the prototype tool

2. THE QUALITY CONTROL PROBLEM

Let A be a finite set of *actions*. We denote by A^* the set of all finite sequences of actions. For a sequence of actions $\alpha = \alpha(1)\dots\alpha(n)$ of length $n = |\alpha|$ we define:

• $\alpha[i,j] = \alpha(i)\dots\alpha(j)$ for any $i \leq j$; otherwise $\alpha[i,j] = \epsilon$

• for $0 \leq i \leq |\alpha|$, ${}^i\alpha = \alpha[1,i]$ (resp. $\alpha^i = \alpha[i,|\alpha|]$) is the prefix (resp. suffix) of length i (resp. $|\alpha| - i + 1$), and ${}^0\alpha = \alpha^{|\alpha|} = \epsilon$

• $set(\alpha)$ is the set consisting of all the elements of α that is, $set(\alpha) = \{\alpha(1), \dots, \alpha(n)\}$.

DEFINITION 1. *For a finite set of actions A, a* **precedence graph** *is a pair $G = (A, \prec)$ where $\prec \subseteq A \times A$ is a partial order. A* **trace** *of G is a sequence of actions $\alpha = \alpha(1)\dots\alpha(n)$ such that:*

• $i \neq j \;\Rightarrow\; \alpha(i) \neq \alpha(j)$

• *for any $1 \leq i \leq |\alpha|$, $set({}^i\alpha)$ is backwards closed that is, $a \in set({}^i\alpha)$ and $a' \prec a$ implies $a' \in set({}^i\alpha)$.*

A trace of length $|A|$ is a **schedule** *of G. We denote by $Sched(G)$ the set of all schedules of G.*

For a precedence graph $G = (A, \prec)$ and a set of actions $B \subseteq A$, we write G/B for the precedence graph restricted to the actions of B that is, $G/B = (B, \prec \cap (B \times B))$. For a trace α, G/α is a notation for $G/set(\alpha)$.

The following definition introduces parameterized systems which are models of an application software with quality

level parameters, worst-case execution times and deadline constraints for its actions. It is parameterized by a function representing the unpredictable, actual execution time.

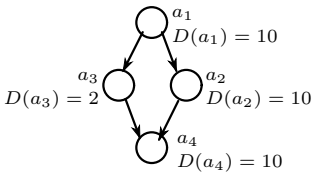

Figure 2: Example of parametarized system.

DEFINITION 2. *A* **parameterized system** $PS(C)$ *is a tuple* $PS(C) = (G, Q, C^{wc}, C, D)$ *where:*
• G *is a precedence graph*
• $Q = [q_{min}, q_{max}]$ *is a finite interval of integers; integers of* Q *correspond to* quality levels
• $C^{wc} : A \times Q \to \mathbb{R}^+$ *($\mathbb{R}^+$ denotes the set of non-negative reals) is a function giving the* worst-case execution time $C^{wc}(a, q)$ *of action* a *for quality level* q. *We assume that, for all* $a \in A$, $q \mapsto C^{wc}(a, q)$ *is a non-decreasing function*
• $C : A \times Q \to \mathbb{R}^+$ *is a parameter, function giving the* actual execution time $C(a, q)$ *of* a *for quality level* q. *We assume that* $C \leq C^{wc}$ *and, for all* $a \in A$, $q \mapsto C(a, q)$ *is a non-decreasing function*
• $D : A \to \mathbb{R}^+$ *is a function giving for any action* a *its* absolute deadline $D(a)$.

A *quality assignment* is a partial function $\theta : A \to Q$ giving for any action a, its quality level $\theta(a)$. We denote by Θ the set of all the quality assignments, and by $\perp$ the quality assignment undefined everywhere. For a quality assignment θ and an action a, $\theta[a \to q]$ is the quality assignment such that, for any $a' \in A$, $a' \neq a$, $\theta[a \to q](a') = \theta(a')$, and $\theta[a \to q](a) = q$.

Functions of the form $C^X : A \times Q \to \mathbb{R}^+$ are called *execution time functions*. We extend execution time functions to obtain partial functions $A^* \times \Theta \to \mathbb{R}^+$. For a sequence of actions α and a quality assignment θ defined on $set(\alpha)$ we take: if $\alpha \neq \epsilon$, then $C^X(\alpha, \theta) = \sum_{1 \leq i \leq |\alpha|} C^X(\alpha(i), \theta(\alpha(i)))$ else $C^X(\epsilon, \theta) = 0$. Notice that $C^X(\alpha, \theta)$ gives the time elapsed when the sequence α is executed with quality θ. We extend any deadline function $D : A \to \mathbb{R}^+$ to non-empty sequences of actions, $D(\alpha) = D(\alpha(|\alpha|))$ that is, $D(\alpha)$ is the deadline of the last action of α.

DEFINITION 3. *Given a parameterized system* $PS(C) = (G, Q, C^{wc}, C, D)$ *we need the following definitions.*
• *A* **state** *is a tuple* (α, θ, t) *such that* α *is a trace of* G, θ *is a quality assignment defined on* $set(\alpha)$ *and* $t = C(\alpha, \theta)$
• *The* **labelled transition relation** $(\alpha, \theta, C(\alpha, \theta)) \xrightarrow{(a,q)} (\alpha', \theta', C(\alpha', \theta'))$ *relates two states* $(\alpha, \theta, C(\alpha, \theta))$ *and* $(\alpha', \theta', C(\alpha', \theta'))$ *if there exists* (a, q) *such that* $\alpha' = \alpha a$ *and* $\theta' = \theta[a \to q]$.
• *An* **execution sequence** *of length* n *is a sequence of consecutive states of the form:*

$$(\epsilon, \perp, 0) \xrightarrow{(a_1,q_1)} (\alpha_1, \theta_1, t_1) \xrightarrow{(a_2,q_2)} \dots \xrightarrow{(a_n,q_n)} (\alpha_n, \theta_n, t_n).$$

• *A* **schedule** *is a pair* (α, θ) *where* $\alpha \in Sched(G)$ *and* θ *is a quality assignment defined on* A.
• *A schedule* α *is* **feasible** *if, for all* k, $C(^k\alpha, \theta) \leq D(^k\alpha)$ *that is, all the actions of* α *complete their execution before their deadline. We denote by* $Sched(PS(C))$ *the set of all the schedules of* $PS(C)$, *and* $Feas(PS(C))$ *the set of all the feasible ones.*
• *The feasibility can be characterizes by a* **schedule function** $t_s : A^* \times \Theta \to \mathbb{R}^+$ *which gives for a schedule* (α, θ), *its* margin *that is, the latest time at which the schedule* (α, θ) *should start in order to meet its deadlines. We take* $t_s(\alpha, \theta) = \mathbf{min}_{1 \leq k \leq |\alpha|}\ t_s(\alpha, \theta)(k)$, *where* $t_s(\alpha, \theta)(k) = D(^k\alpha) - C(^k\alpha, \theta)$. *Then, the feasibility of* (α, θ) *can be characterized by* $0 \leq t_s(\alpha, \theta)$.

Consider a parameterized system $PS(C) = (G, Q, C, C^{wc}, D)$ with four actions $A = \{a_1, a_2, a_3, a_4\}$ and two quality levels $Q = \{0, 1\}$, such that G and D are given in figure 2. The sequences $\alpha_1 = a_1a_2a_3a_4$ and $\alpha_2 = a_1a_3a_2a_4$ are schedules of G. Let θ_0 be the constant quality assignment $\theta_0 = 0$. Notice that, if $C(a_i, q) = C^{wc}(a_i, q) = 2q + 1$ for all a_i, then the schedule (α_1, θ_0) is not feasible, whereas the schedule (α_2, θ_0) is feasible.

DEFINITION 4. *Given a parameterized system* $PS(C)$, *a* **controller** Γ *is a function* $\Gamma(\alpha, \theta, t) = (a, q)$ *such that* αa *is a trace of* G *and* $q \in Q$.
We denote by $PS(C)||\Gamma$ *the* controlled parameterized system *characterized by the labelled transition relation* $(\alpha, \theta, t) \xrightarrow{(a,q)} (\alpha', \theta', t')$ *such that* (α, θ, t) *and* (α', θ', t') *are states of* $PS(C)$, *and* $(a, q) = \Gamma(\alpha, \theta, t)$.

Notice that a controller determinizes the behavior of $PS(C)$ by selecting, for each state (α, θ, t), the next action to be executed and its quality level. Thus, for a given C, it computes a schedule (α, θ).

DEFINITION 5 (QUALITY CONTROL PROBLEM). *Given a parameterized system* $PS(C) = (G, Q, C^{wc}, C, D)$, *find a controller* Γ *such that, for any actual execution time function* C, *it computes an* **optimal feasible schedule** (α, θ) *that is:*
• *the overall execution time* $C(\alpha, \theta)$ *is maximal with respect to all feasible schedules of* $PS(C)$ *that is,* $C(\alpha, \theta) = \mathbf{max}\ \{\ C(\alpha', \theta') \mid (\alpha', \theta') \in Feas(PS(C))\ \}$
• (α, θ) *is a feasible schedule of* $PS(C)$.

In addition to feasibility and optimality, we require *smoothness* for the computed schedules. Informally, for a schedule (α, θ), smoothness means low fluctuation of the quality assignment θ along the sequence α. We do not formalize this property which is essential for most multimedia applications.

3. THE ABSTRACT CONTROL ALGORITHM

In the rest of the paper, $PS(C)$ is the parameterized system $PS(C) = (G, Q, C^{wc}, C, D)$.

3.1 Quality Control for Known C

We provide preliminary results about the quality control problem when C is known. In that case, C^{wc} is not useful for computing feasible schedules.

DEFINITION 6. *Given* $PS(C)$, *we denote by* $Best\_Sched$ *any function which, for any quality assignment* θ, *gives the*

schedule $\alpha_\theta = Best\_Sched(\theta)$ of G such that (α_θ, θ) maximizes the schedule function t_s that is:

$$t_s(\alpha_\theta, \theta) = \mathbf{max}\ \{\ t_s(\alpha, \theta) \mid (\alpha, \theta) \in Sched(PS(C))\ \}.$$

Notice that, for any schedule (α, θ) of $PS(C)$, the overall execution time $C(\alpha, \theta)$ is independent of α. We define the binary relation $\ll$ on Θ by $\theta \ll \theta' \Leftrightarrow C(\alpha, \theta) < C(\alpha, \theta')$ for each schedule α. The relation $\ll$ is a strict total order on classes of quality assignments $\{\ \theta \mid C(\alpha, \theta) = constant\ \}$.

PROPOSITION 1. *Given $PS(C)$, an optimal feasible schedule $(\alpha_{\theta_M}, \theta_M)$ is computed by:*
for all $\theta \in \Theta$ **do** $\alpha_\theta := Best\_Sched(\theta)$ **od**
$\theta_M := \mathbf{max}_{\ll}\ \{\ \theta \mid 0 \leq t_s(\alpha_\theta, \theta)\ \}$
return $(\alpha_{\theta_M}, \theta_M)$
where $\mathbf{max}_{\ll}$ is the function giving an element of the maximal class of its arguments according to $\ll$.

Proof: Assume that there exists (α, θ) such that (α, θ) is feasible and $\theta_M \ll \theta$. Then $0 \leq t_s(\alpha, \theta) \leq t_s(\alpha_\theta, \theta)$ because $\alpha_\theta = Best\_Sched(\theta)$ maximizes t_s. We conclude that (α_θ, θ) is a feasible schedule, so $\theta \ll \theta_M$. (Contradiction). □

The following results can be used to reduce the computation of *Best_Sched* to the computation of EDF schedules for G.

DEFINITION 7 (EDF SCHEDULE). *Given $PS(C)$ with a precedence graph G and a deadline function D, we denote by D^* the deadline function:*

$$D^*(a) = \mathbf{min}\ \{\ D(a') \mid a \prec a'\ \}.$$

We say that a schedule α of G is an **EDF** *schedule if, for all $i < j$, $D^*({}^i\alpha) \leq D^*({}^j\alpha)$. We denote by $EDF(G, D)$ the set of the EDF schedules of $PS(C)$.*

Notice that $Feas(PS(C)) = Feas(PS^*(C))$ where $PS^*(C) = (G, Q, C^{wc}, C, D^*)$ as both $PS(C)$ and $PS^*(C)$ have the same set of critical deadlines.

PROPOSITION 2. *For a quality assignment θ, the EDF schedules maximize $t_s(\alpha, \theta)$ that is, for any schedule α, there exists an EDF schedule α' such that $t_s(\alpha, \theta) \leq t_s(\alpha', \theta)$.*

The above proposition allows to compute, for known C, an optimal feasible schedule (α, θ_M), by statically computing an EDF schedule α and checking its feasibility for the different quality assignments θ.

LEMMA 1. *Let α be a schedule such that there exist two consecutive and independent actions $a = \alpha(i)$ and $b = \alpha(i+1)$ (two actions a and b are independent if $a \not\prec b$ and $b \not\prec a$), such that their deadlines are inverted that is, $D(a) \geq D(b)$. For the schedule α' in which a and b are swapped that is, $\alpha'(j) = \alpha(j)$ for all $j \notin \{i, i+1\}$, $\alpha'(i) = b$ and $\alpha'(i+1) = a$, we have:*

$$t_s(\alpha', \theta) \geq t_s(\alpha, \theta).$$

Proof of proposition 2: We apply lemma 1 as follows. Let α be an arbitrary schedule, and α^{EDF} be an EDF schedule. It is possible to obtain α^{EDF} from α by commuting successively two consecutive independent actions with inverted deadlines D^*. Thus, EDF schedules maximize $\mathbf{min}_{1 \leq k \leq n}\ D^*(\alpha^k) - C({}^k\alpha, \theta) = \mathbf{min}_{1 \leq k \leq n}\ D({}^k\alpha) - C({}^k\alpha, \theta) = t_s(\alpha, \theta)$. □

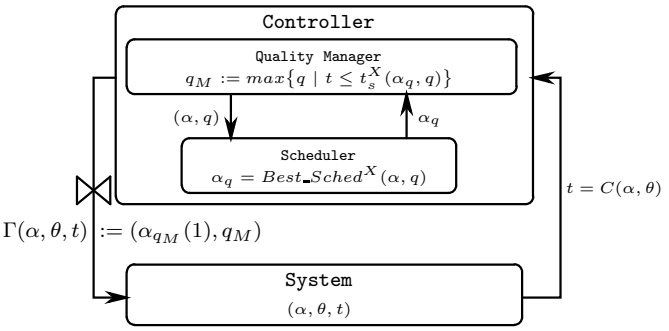

Figure 3: Controller's architecture

3.2 Quality Control under Uncertainty

We propose a control algorithm allowing the incremental online computation of schedules (α, θ) of $PS(C)$. The control algorithm is a dynamic adaptation of the proposition 1. It uses appropriate approximations t_s^X of t_s, and $Best\_Sched^X$ of $Best\_Sched$.

DEFINITION 8. *For an execution time function $C^X : A^* \times \Theta \to \mathbb{R}^+$, we define the corresponding schedule function t_s^X:*

$$\begin{aligned} t_s^X(\alpha, \theta)(k) &= D({}^k\alpha) - C^X({}^k\alpha, \theta) \\ t_s^X(\alpha, \theta) &= \mathbf{min}_{1 \leq k \leq |\alpha|}\ t_s^X(\alpha, \theta)(k). \end{aligned}$$

This definition of t_s^X generalizes definition 3 of t_s. In the proposed control algorithm, functions $Best\_Sched^X$ will be used in a similar manner as functions $Best\_Sched$ in algorithm 1. Nevertheless, $Best\_Sched$ and $Best\_Sched^X$ do not have the same profile. We denote by $Best\_Sched^X$ a function which, for any trace α and quality level q, gives a sequence $\alpha_q = Best\_Sched^X(\alpha, q)$ such that $\alpha\alpha_q$ is a schedule of G. The functions $Best\_Sched^X$ will be chosen so as to maximize t_s^X. Finding adequate functions $Best\_Sched^X$ is a non trivial problem discussed in 5.2.

DEFINITION 9 (ABSTRACT CONTROL ALGORITHM). *Given $PS(C)$, a schedule function t_s^X and a function $Best\_Sched^X$, we define the controller Γ such that, for any state (α, θ, t), $\Gamma(\alpha, \theta, t)$ is computed as follows:*
for all $q \in Q$ **do** $\alpha_q := Best\_Sched^X(\alpha, q)$ **od**
$q_M = \mathbf{max}\ \{\ q \mid t \leq t_s^X(\alpha_q, q)\ \}$
return $\Gamma(\alpha, \theta, t) := (\alpha_{q_M}(1), q_M)$.

Figure 3 shows the controller's architecture. It is composed of a Scheduler computing $Best\_Sched^X$, and of a Quality Manager applying the *quality management policy* $t \leq t_s^X$. From a given state (α, θ, t),
• The Scheduler computes for each quality level $q \in Q$, a sequence of actions $\alpha_q = Best\_Sched^X(\alpha, q)$. The pair (α_q, q) is a schedule for the remaining actions.
• The Quality Manager computes the maximal quality level q_M meeting the quality management policy that is,

$$q_M = \mathbf{max}\ \{\ q \mid t \leq t_s^X(\alpha_q, q)\ \}.$$

In the rest of the paper, we present theoretical and experimental results concerning the choice of t_s^X and $Best\_Sched^X$ for the controller to compute feasible, near-optimal and smooth schedules. In the following section, we propose quality management policies for safety and smoothness.

4. QUALITY MANAGEMENT POLICIES

4.1 Quality Management Policies for Safety

DEFINITION 10. *Given $PS(C)$, we introduce the* safe *schedule function t_s^{sf} corresponding to the* safe *execution time function $C^{sf} : A^* \times \Theta \to \mathbb{R}^+$ defined by:*

$$C^{sf}(\alpha, \theta) = C^{wc}(\alpha(1), \theta) + C^{wc}(\alpha^2, q_{min}).$$

Notice that $t \leq t_s^{sf}(\alpha, q)$ guarantees feasibility for executing the first action $\alpha(1)$ of α with quality q and the rest of the actions of α with quality q_{min} under the worst-case assumption. This implies that the schedule $(\alpha, \theta[\alpha(i) \to q_{min}, i > 1])$ is feasible for any C.

PROPOSITION 3 (SAFETY). *Consider the parameterized system $PS(C)$ and a schedule α_0 such that (α_0, q_{min}) is a feasible schedule of $PS(C^{wc})$. For any controller with quality management policy $t \leq t_s^{sf}$, the computed schedules are feasible if $Best\_Sched^{sf}$ gives for minimal quality EDF schedules that is, for any α, $\alpha_{q_{min}} = Best\_Sched^{sf}(\alpha, q_{min}) \in EDF(G/\alpha_{q_{min}}, D)$.*

It can be shown that the above proposition also holds when t_s^{sf} is replaced by any schedule function t_s^X such that $t_s^X \leq t_s^{sf}$ with $t_s^X(\alpha, q_{min}) = t_s^{sf}(\alpha, q_{min})$, and $Best\_Sched^X$ meets the same properties as $Best\_Sched^{sf}$. This result is used in the next section to find other quality management policies that are not only safe but also smooth.

LEMMA 2. *For any controller Γ satisfying the assumptions of proposition 3, we have $q_{min} \in \{q \mid t \leq t_s^{sf}(\alpha_q, q)\}$ at any reachable state (α, θ, t) of $PS(C)||\Gamma$, where $\alpha_q = Best\_Sched^{sf}(\alpha, q)$.*

Proof of proposition 3: By the lemma 2, we have $q_{min} \in \{q \mid t \leq t_s^{sf}(\alpha_q, q)\}$ at any state of $PS(C)||\Gamma$ that is, $\{q \mid t \leq t_s^{sf}(\alpha_q, q)\}$ is a non-empty set. Let (α, θ, t) and (α', θ', t') be two states of $PS(C)||\Gamma$ such that $(\alpha, \theta, t) \stackrel{(a, q_M)}{\longrightarrow} (\alpha', \theta', t')$, and let $\alpha_q = Best\_Sched^{sf}(\alpha, q)$ be the planed schedule for the quality level q. Then we have $t \leq t_s^{sf}(\alpha_{q_M}, q_M)$ and the action $a = \alpha_{q_M}(1)$ meets its deadline:

$$\begin{aligned} & t \leq t_s^{sf}(\alpha_{q_M}, q_M) = \mathbf{min}_{1 \leq k \leq |\alpha_{q_M}|}\ t_s^{sf}(\alpha_{q_M}, q_M)(k) \\ \Rightarrow\ & t \leq t_s^{sf}(\alpha_{q_M}, q_M)(1) \\ \Rightarrow\ & t \leq D(a) - C^{wc}(a, q_M) \\ \Rightarrow\ & t' \leq t + C^{wc}(a, q_M) \leq D(a). \end{aligned}$$

This demonstrates that any action a meets its deadline that is, the schedule (α, θ) is feasible. □

4.2 Improving Smoothness

We discuss the influence of three quality management policies on quality smoothness. The influence is illustrated by an example and confirmed by experimental results in section 6.4. It is not treated formally as we do not have yet a completely worked out framework.

4.2.1 Simple Quality Management Policy

Consider $PS(C)$ with three actions, four quality levels $Q = \{1, \ldots, 4\}$, and a single deadline $D = 9$ (for all i, $D(\alpha(i)) = 9$). We assume that $PS(C)$ has a schedule α such that the actual and the worst-case execution times are identical for all actions, as given in the table of figure 4. The computed quality assignment for α by using the safe policy $t \leq t_s^{sf}$ is not smooth (figure 4). We can improve smoothness by combining worst-case and average behavior. We define the *average* execution time function $C^{av} : A \times Q \to \mathbb{R}^+$. Average execution times can be estimated by static analysis and/or profiling techniques. Denote by t_s^{av} the corresponding schedule function.

In [6], we define the *simple* quality management policy $t \leq t_s^{sp}$, where $t_s^{sp} = \mathbf{min}\ \{\ t_s^{sf}, t_s^{av}\ \}$. Notice that using t_s^{sp} also leads to feasible schedules. For the previous example, the schedule computed by using $t \leq t_s^{sp}$ is smoother than the one computed by using $t \leq t_s^{sf}$ (figure 4).

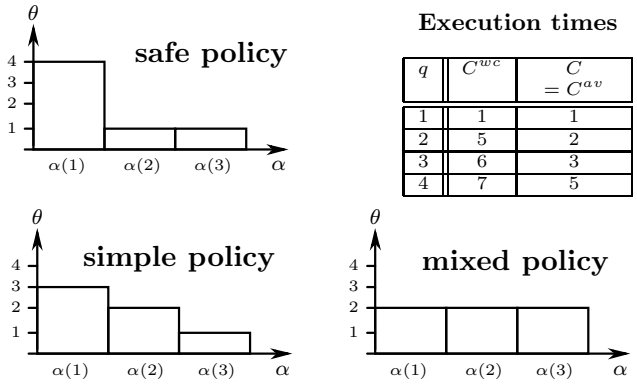

Figure 4: Comparison between different policies

4.2.2 Mixed Quality Management Policy

The simple management policy may lead to a drastical reduction of the quality before a critical deadline. Even if actual time follows exactly average time (i.e $C = C^{av}$), the quality may need to be decreased along a sequence of actions where $C^{sf} > C^{av}$. We propose the *mixed* quality management policy which is robust with respect to this phenomena.

We define the function $\delta = C^{sf} - C^{av}$. Let $\delta^{max}(\alpha, \theta) = \mathbf{max}_{i \geq 1}\ \delta(\alpha^i, \theta)$. Given (α, θ), the value $\delta^{max}(\alpha, \theta)$ characterizes the difference between worst-case and average behavior. The schedule function t_s^{mx} corresponds to the *mixed* execution time function C^{mx} defined by $C^{mx} = C^{av} + \delta^{max}$. For the example given in figure 4, the schedule computed by using the mixed policy $t \leq t_s^{mx}$ is the smoothest one (θ is constant).

5. RESULTS FOR OPTIMALITY

As the functions t_s^X and $Best\_Sched^X$ approximate t_s and $Best\_Sched$, the computed schedules may not be optimal.

5.1 Conditions for Optimality

We provide bounds relating actual execution time for completing a sequence of actions α and the corresponding deadline $D(\alpha)$, which is the available time budget, for two different quality management policies:

• The average policy $t \leq t_s^{av}$ which does not take into account worst-case behavior. For this case, we show that if $C = C^{av}$ the difference between the time for completing α and the corresponding deadline $D(\alpha)$ is less than a constant Δ characterizing the granularity of control.

• The mixed policy $t \leq t_s^{mx}$ which jointly takes into account average and worst-case behavior. For this case, we show that if $C = C^{av}$ the difference between the time for com-

pleting α and the corresponding deadline $D(\alpha)$ is less than a constant depending on Δ and δ^{max}.

5.1.1 Average Behavior – Speed Diagram for $t \leq t_s^{av}$

Consider $PS(C)$ with a controller applying the quality management policy $t \leq t_s^{av}$. Let (α, θ, t) be a state of the $PS(C)||\Gamma$, and (α_q, q) be the planed schedule from this state. We show that the quality management policy $t \leq t_s^{av}$ admits a geometric interpretation in terms of relative speeds between actual and average execution time.

We introduce the following abbreviations for $1 \leq k \leq |\alpha_q|$: $c = C^{av}(\alpha, \theta)$, $d_k(q) = D({}^k\alpha_q)$ and $c_k(q) = c + C^{av}({}^k\alpha_q, q)$. We represent system's evolution in a two-dimensional space (figure 5) with coordinates t (actual time) and $y(q)$ the *normalized average time* with respect to the deadline d_k defined by:

$$y(q) = \frac{c}{c_k(q)} \cdot d_k(q).$$

As a result of the normalization, points on the diagonal (45 degree slope) correspond to optimal behavior. Points below the diagonal correspond to states where the actual computation is late with respect to average time. For points above the diagonal, the computation goes faster than estimated.

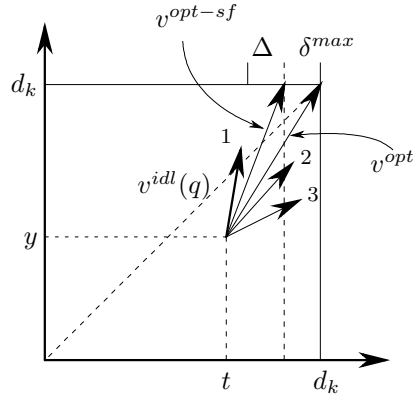

Figure 5: Speed diagram

We introduce two notions of speed (ratio $y(q)/t$) to explain the quality management policy $t \leq t_s^{av}$ of the controller.

• The *ideal speed* $v^{idl}(q)$ which is the speed when $C = C^{av}$ that is, in the ideal case where the actual time is equal to the average time:

$$v^{idl}(q) = \frac{y(q)}{t} = \frac{d_k(q)}{c_k(q)}.$$

• The *optimal speed* $v^{opt}(q)$ at point $(t, y(q))$ is equal to the slope of the segment from this point to $(d_k(q), d_k(q))$ that is, this is the slope allowing to reach the deadline when it expires:

$$v^{opt}(q) = \frac{d_k(q) - y(q)}{d_k(q) - t} = \frac{d_k(q)}{c_k(q)} \cdot \frac{c_k(q) - c}{d_k(q) - t}.$$

The optimal speed $v^{opt}(q)$ corresponds to an optimal behavior of the system in which the action $\alpha_q(k)$ completes exactly at $d_k(q)$. Figure 5 shows an example with three quality levels in which $y(q)$ and $v^{opt}(q)$ do not depend on q.

PROPOSITION 4. *Let (α, θ, t) be a state of $PS(C)||\Gamma$ with the quality management policy $t \leq t_s^{av}$. Given a quality level q, a sequence α_q and an index k, we have:*

$$v^{idl}(q) \geq v^{opt}(q) \iff t \leq t_s^{av}(\alpha_q, q)(k).$$

Proof:

$$\begin{aligned} & v^{opt}(q) \leq v^{idl}(q) \\ \Leftrightarrow\ & \frac{c_k(q) - c(q)}{d_k(q) - t} \leq 1 \\ \Leftrightarrow\ & t \leq d_k(q) - (c_k(q) - c(q)) \\ \Leftrightarrow\ & t \leq D({}^k\alpha_q) - C^{av}({}^k\alpha_q, q) \\ \Leftrightarrow\ & t \leq t_s^{av}(\alpha_q, q)(k). \quad \square \end{aligned}$$

In the example of figure 5, the application of the average quality management policy $t \leq t_s^{av}$ amounts to choosing quality corresponding to the speed $v^{idl}(1)$ which is the best quality such that $v^{idl}(q) \geq v^{opt}(q)$.

In the ideal case where $C = C^{av}$, the average policy $t \leq t_s^{av}$ is sufficient to ensure feasibility of the computed schedules. In this case, the next proposition explains that the overall execution time can be estimated with a precision of Δ, where Δ depends on the granularity of control:

$$\Delta = \mathbf{max}\ \{\ C^{av}(a, q+1) - C^{av}(a, q) \mid q < q_{max}, a \in A\ \}.$$

PROPOSITION 5. *Consider $PS(C)||\Gamma$ with the quality management policy $t \leq t_s^{av}$, and a single deadline D (constant deadline function). If (α, θ) is a schedule computed by the controller such that $\theta(\alpha(1)) < q_{max}$ and $C = C^{av}$, then:*

$$D - \Delta \leq C(\alpha, \theta) \leq D.$$

5.1.2 Worst-Case and Average Behavior – Speed Diagram for $t \leq t_s^{mx}$

In order to take into account average and worst-case behavior, a similar construction of a speed diagram can be carried out for the mixed quality management policy $t \leq t_s^{mx}$.

Consider $PS(C)$ with a controller applying the quality management policy $t \leq t_s^{mx}$. Let (α, θ, t) be a state of $PS(C)||\Gamma$, and (α_q, q) be the planed schedule from this state. If $(t, y(q))$ is the point of the speed diagram corresponding to (α, θ, t), a notion of *safe* optimal speed can be defined as follows:

$$v^{opt-sf}(q) = \frac{d_k(q)}{c_k(q)} \cdot \frac{c_k(q) - c(q)}{d_k(q) - \delta^{max}(q) - t}.$$

where $\delta^{max}(q)$ is an abbreviation for $\delta^{max}({}^k\alpha_q, q)$. As shown in figure 5, $v^{opt-sf}(q)$ is the slope of the segment from the considered point $(t, y(q))$ to $(d_k(q) - \delta^{max}(q), d_k(q))$. The difference from the previous case comes from the requirement for feasibility with respect to the deadline d_k. By targeting point $(d_k(q) - \delta^{max}(q), d_k(q))$ the controller respects a safety margin $\delta^{max}(q)$ which is sufficient to ensure termination before the deadline d_k. This constant $\delta^{max}(q)$ characterizes a tradeoff between feasibility and optimality.

PROPOSITION 6. *Let (α, θ, t) be a state of $PS(C)||\Gamma$ with the quality management policy $t \leq t_s^{mx}$. Given a quality level q, a sequence α_q and an index k, we have:*

$$v^{idl}(q) \geq v^{opt-sf}(q) \iff t \leq t_s^{mx}(\alpha_q, q)(k).$$

Proof:

$$\begin{aligned} & v^{opt}(q) \leq v^{idl-sf}(q) \\ \Leftrightarrow\ & \frac{c_k(q) - c(q)}{d_k(q) - \delta^{max}(q) - t} \leq 1 \\ \Leftrightarrow\ & t \leq d_k(q) - \delta^{max}(q) - (c_k(q) - c(q)) \\ \Leftrightarrow\ & t \leq D({}^k\alpha_q) - C^{av}({}^k\alpha_q, q) - \delta^{max}({}^k\alpha_q, q) \\ \Leftrightarrow\ & t \leq t_s^{mx}(\alpha_q, q)(k). \quad \square \end{aligned}$$

When $C = C^{av}$ and the controller uses the schedule function t_s^{av}, the proposition 5 gives an estimate of the overall execution with a precision of Δ. The next proposition establishes that, when using t_s^{mx} instead of t_s^{av}, this precision becomes $\Delta + \delta^{max}(\alpha, q+1)$, where α is the schedule computed by the controller and q the quality level of the first action $\alpha(1)$.

PROPOSITION 7. *Consider $PS(C)||\Gamma$ with the mixed quality management policy $t \leq t_s^{mx}$, and a function $Best\_Sched^{mx}$ such that it returns schedules that maximize t_s^{mx}. For a single deadline D, if (α, θ) is a schedule computed by the controller such that $q = \theta(\alpha(1)) < q_{max}$ and $C = C^{av}$, we have:*

$$D - \Delta - \delta^{max}(\alpha, q+1) \leq C(\alpha, \theta) \leq D.$$

5.2 Improving $Best\_Sched^{mx}$

A crucial question for optimality is computing efficiently schedules that maximize the schedule function for a considered quality management policy. Contrary to the results of section 3.1 showing that arbitrary EDF schedules maximize t_s (for known C), it is easy to see that this is not true for quality management policies taking into account worst-case behavior. To illustrate this fact, consider an example for three actions $A = \{a_1, a_2, a_3\}$, two quality levels $Q = \{1, 2\}$, and a single deadline $D = 30$. Assume that the precedence graph is empty ($G = (A, \emptyset)$), and that execution times are such that, $C^{av}(a_i, q) = 4q$ for all i, $C^{wc}(a_i, q) = 4q$ for $i \in \{1, 2\}$, and $C^{wc}(a_3, q) = 20$. The schedules $\alpha_1 = a_1a_2a_3$ and $\alpha_2 = a_3a_1a_2$ lead to different values $t_s^{mx}(\alpha_1, 2) = -6 < t_s^{mx}(\alpha_2, 2) = 2$. Thus, α_2 is a better schedule than α_1 with respect to the mixed policy $t \leq t_s^{mx}$. This section provides results for selecting amongst the EDF schedules, the ones which maximize the mixed scheduling function t_s^{mx}.

DEFINITION 11. *We say that an EDF schedule α of G is* **EDF-optimal** *with respect to the quality level q if:*

$$t_s^{mx}(\alpha, q) = \mathbf{max}\ \{\ t_s^{mx}(\alpha', q) \mid \alpha' \in EDF(G, D)\ \}.$$

PROPOSITION 8. *The deadline function D^* defined in section 3.1 induces a partition $A_1 \dots A_L$ such that $D^*(A_1) < \dots < D^*(A_L)$. Any EDF schedule α can be written as $\alpha = \alpha_1 \dots \alpha_L$, where $\alpha_l \in Sched(G/A_l)$. Then we have:*
- *If $\delta^{max}(\alpha_l, q)$ is minimal, then α_l is an EDF-optimal schedule of G/A_l with respect to the quality level q.*
- *If for all l, α_l is EDF-optimal, then α is EDF-optimal.*

For a given quality level q, the above proposition allows to compute an EDF-optimal schedule by concatenation of EDF-optimal schedules α_l. For each schedule α_l, EDF-optimality can be achieved by a minimization of δ^{max}. Proposition 9 gives three transformation rules that can be used to decrease $\delta^{max}(\alpha_l, q)$. These rules rely on two characteristic values: $\eta(a, q) = C^{wc}(a, q) - C^{av}(a, q)$, which is the "uncertainty" of a for quality q, and $\beta(a, q) = C^{wc}(a, q_{min}) - C^{av}(a, q)$, which is related to the "fall-back ability" of a for quality q.

PROPOSITION 9 (MINIMIZING δ^{max}). *Given a schedule α and a quality level q, consider two consecutive actions $a = \alpha(i)$ and $b = \alpha(i+1)$ such that $a \not\prec b$. Let α' be the schedule in which a and b are swapped that is, $\alpha'(j) = \alpha(j)$ for $j \notin \{i, i+1\}$, $\alpha'(i) = b$ and $\alpha'(i+1) = a$. Then, we have $\delta^{max}(\alpha', q) \leq \delta^{max}(\alpha, q)$ in the following situations:*
R1: *$\eta(a, q) < \eta(b, q)$, $\beta(a, q) \leq 0$ and $\beta(b, q) \leq 0$*
R2: *$\beta(a, q) \leq 0$ and $\beta(b, q) > 0$.*
R3: *$(\eta - \beta)(a, q) > (\eta - \beta)(b, q)$, $\beta(a, q) > 0$, and $\beta(b, q) > 0$.*

For a given quality level q, the conditions R1, R2, R3 define rules for getting, from an EDF schedule α, an EDF schedule α' such that $t_s^{mx}(\alpha', q) \geq t_s^{mx}(\alpha, q)$.

6. EXPERIMENTAL RESULTS

6.1 The Context – Tools and Case Study

We applied these results to an MPEG 4 encoder provided by STMicroelectronics and written in C (more than 7000 loc). The encoder treats frames cyclically. Each frame is split into N ($396 \leq N \leq 1620$) macroblocks of 256 pixels.

We developed a prototype tool allowing the generation of controlled application software (figure 1). The inputs of the tool are:
- the precedence graph G corresponding to the treatment of a macroblock and its iteration parameter N;
- tables describing the functions D, C^{av} and C^{wc} for the actions of G.

From these inputs the tool computes:
- C code corresponding to statically precomputed schedules;
- tables containing pre-computed values used by the controller for the computation of the chosen quality management policy.

A compiler is used to link the following items and to generate the controlled application software from:
- the schedule and the tables generated by the tool;
- application code for the actions of the schedule;
- a generic controller mainly consisting of a quality manager.

The overhead due to the instrumentation of the application software in the size of the compiled code, memory allocation, and overall execution time, is less than 2 %. For all these estimates, we assume that the application software runs on a single processor without OS and that it is possible to read a register counting the number of CPU cycles elapsed. We provide experimental results for a platform consisting of a single XiRisc processor [1]. The platform is simulated by using the eliXim tool of STMicroelectronics. We considered a benchmark of 582 frames, consisting of 9 sequences produced by a camera every $P = 40\ ms$ (single deadline $D = 40\ ms$).

6.2 Controlled Quality vs Constant Quality

The overall execution time is shown in figure 6, for controlled quality and constant quality $q = 3$. In the latter case, an input buffer of size 2 allows a maximal latency of 80 ms (this corresponds to standard industrial practice). Notice the presence in figure 6 of two kinds of jumps: eight jumps corresponding to changes of video sequences (encoding of I-frames); two bursts of jumps corresponding to frame skips due to buffer overflow occurring for constant quality only.

Experimental results show that for constant quality level, load fluctuation leads to a high amount of frame skips even if an additional input buffer is used. On the contrary, controlled quality completely avoids frame skips and overloads lead to smooth reduction of the video quality.

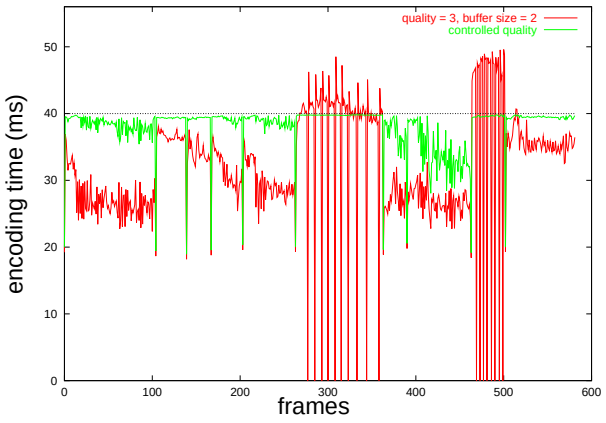

Figure 6: Controlled quality vs constant quality

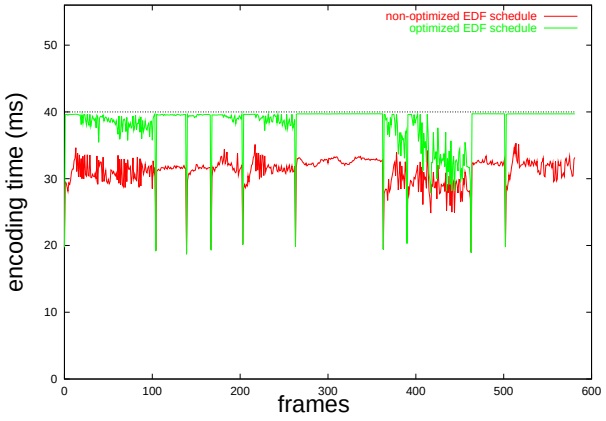

Figure 7: Optimization of $Best\_Sched^{mx}$

6.3 Using Optimized Schedules

We have compared the controlled application running with a non-optimized function $Best\_Sched^{mx}$ (high values of δ^{max}), and the same controlled application running with a function $Best\_Sched^{mx}$ obtained by applying the three optimization rules given in 5.2.

Figure 7 shows that the overall execution time is close to the deadline of $D = 40\ ms$ when $Best\_Sched^{mx}$ is optimized, whereas approximately 7 ms are lost in average when $Best\_Sched^{mx}$ is non-optimized. This corresponds to the difference of the values δ^{max} encountered during the execution, between the optimized and the non-optimized case (7.2 ms in average).

6.4 The Impact of Quality Management Policy

We have compared safe, simple and mixed quality management policies. We provide results for a static schedule α. We build the speed diagram (see figure 8) for a particular input frame.

For safe and simple quality management policies significant speed discontinuities are observed at points A, B, C, D and E. For the safe policy (resp. simple policy), the speed of the system from point A (resp. B) to point C corresponds to a choice of minimal quality. This drastically reduces the video quality. On the contrary, for mixed policy we get almost uniformly constant speed which leads to significantly improved video quality.

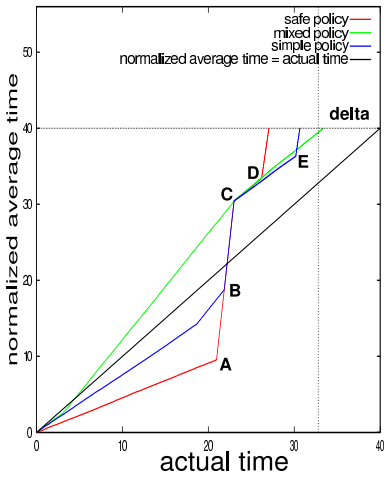

Figure 8: Speed diagram for different policies

7. CONCLUSION

The presented method uses fine grain control to meet safety and best effort requirements. It overcomes limitations of hard real-time approaches where strict respect of deadlines implies poor time budget utilization. This is possible through fine grain control, which allows adaptation to load changes during a cycle instead of using a priori known global execution time estimates.

The method is founded on a solid theoretical basis. The controller computes feasible schedules under reasonable assumptions about the controlled system — the existence of statically computable feasible schedules for minimal quality. The behavior of the controller is characterized by its quality management policy and the associated $Best\_Sched$ function. Experimental results show that mixed quality management policy significantly improves quality smoothness with respect to the two other policies. Speed diagrams provide a general framework for analyzing the controller's performance, admitting an intuitive geometric interpretation. The results show that even in the ideal case where actual execution times agree with average execution times, meeting safety requirements inherently limits achieving optimality. The actual execution time may not fill the entire available time budget. The amount of the available time which is lost must be lower than a constant, which depends on the difference between average and worst-case behavior as well as granularity of control . The way this constant increases when actual and average execution time differ is the subject of ongoing work. Finally, an important finding is that all EDF schedules are not equivalent with respect to the considered quality management policies. The rules provided for computing the schedules which maximize the corresponding schedule functions, define strategies for improving predictability in the presence of uncertainty.

Experimental results confirm the interest of the method and its low overhead. Given their importance, we actively work in several directions to improve the prototype tool: compositional generation of EDF schedules for iterative programs, efficient computation of $Best\_Sched$ functions, and application of learning techniques for better estimation of the average execution times.

8. REFERENCES

[1] http://xirisc.deis.unibo.it/.

[2] K.-E. Arzen, B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson, and L. Sha. Integrated control and scheduling. Technical report.

[3] 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.

[4] 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.

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

[6] 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.

[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.

[8] D. Isovic, G. Fohler, and L. Steffens. Timing constraints of mpeg-2 decoding for high quality video: misconceptions and realistic assumptions.

[9] G. Koren and D. Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. Technical Report TR1996-715, , 1996.

[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*.

[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] L. Papalau, C. M. O. Pérez, and L. Steffens. In S. Goddard, editor, *Work-In-Progress Session of the 16th Euromicro Conference on Real-Time Systems*, pages 33–36, 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.

9. APPENDIX

9.1 Lemma 1

Proof of lemma 1: From the definition of t_s, we have:

$$t_s(\alpha,\theta)(i) \geq t_s(\alpha,\theta)(i+1). \quad (1)$$

We have $t_s(\alpha',\theta)(j) = t_s(\alpha,\theta)(j)$ for $j \neq i, i+1$, and:

$$t_s(\alpha',\theta)(i) \geq t_s(\alpha,\theta)(i+1) \quad (2)$$

$$t_s(\alpha',\theta)(i+1) \geq t_s(\alpha,\theta)(i+1). \quad (3)$$

We conclude from (1), (2), (3), that $t_s(\alpha',\theta) \geq t_s(\alpha,\theta)$. □

9.2 Lemma 2

LEMMA 3. *For any (α,θ), we have:*

$$t_s^{sf}(\alpha,\theta) \leq t_s^{sf}(\alpha^2, q_{min}) - C^{wc}(\alpha(1),\theta).$$

Proof of lemma 3: We have:

$$t_s^{sf}(\alpha,\theta) \leq \mathbf{min}_{2\leq k\leq|\alpha|}\ t_s^{sf}(\alpha,\theta)(k) \quad (4)$$

where $t_s^{sf}(\alpha,\theta)(k) = D(^k\alpha) - C^{sf}(^k\alpha,\theta)$

$$= D(^k\alpha) - C^{wc}(\alpha(1),\theta) - C^{wc}(\alpha[2,k], q_{min}). \quad (5)$$

For any $k \geq 2$, we can rewrite (5) as:

$$\begin{aligned} & D(^{k-1}(\alpha^2)) - C^{wc}(^{k-1}(\alpha^2), q_{min}) - C^{wc}(\alpha(1),\theta) \\ = \ & t_s^{wc}(\alpha^2, q_{min})(k-1) - C^{wc}(\alpha(1),\theta) \\ = \ & t_s^{sf}(\alpha^2, q_{min})(k-1) - C^{wc}(\alpha(1),\theta). \end{aligned} \quad (6)$$

From (4), (6), $t_s^{sf}(\alpha,\theta) \leq t_s^{sf}(\alpha^2, q_{min}) - C^{wc}(\alpha(1),\theta)$. □

Proof of lemma 2: The proof is made by induction on the reachable states.

• **initialization:** state $(\epsilon, \perp, 0)$

As there exists α_0 such that (α_0, q_{min}) is a feasible schedule of $PS(C^{wc})$, we have:

$$0 \leq t_s^{wc}(\alpha_0, q_{min}) = t_s^{sf}(\alpha_0, q_{min}).$$

Let $\alpha_{q_{min}} = Best\_Sched^{sf}(\epsilon, q_{min})$ be the planed schedule for the lowest quality at initialization. It can be shown that results of proposition 2 also hold for t_s^{sf} and q_{min}. As $\alpha_{q_{min}}$ is an EDF schedule of G and α_0 is a schedule of G, by proposition 2 we obtain:

$$0 \leq t_s^{sf}(\alpha_0, q_{min}) \leq t_s^{sf}(\alpha_{q_{min}}, q_{min}).$$

• **induction:** $(\alpha,\theta,t) \xrightarrow{(a,q_M)} (\alpha',\theta',t')$

Let (α,θ,t) be a reachable state of $PS(C)||\Gamma$, and $(a,q_M) = \Gamma(\alpha,\theta,t)$. Let (α',θ',t') be a state of $PS(C)||\Gamma$ such that $(\alpha,\theta,t) \xrightarrow{(a,q_M)} (\alpha',\theta',t')$, $\alpha_q = Best\_Sched^{sf}(\alpha,q)$, and $\alpha'_q = Best\_Sched^{sf}(\alpha',q)$. Let $q_M = \mathbf{max}\{q \mid t \leq t_s^{sf}(\alpha_q,q)\}$, and $a = \alpha_{q_M}(1)$.

By induction hypothesis, we have $q_{min} \in \{q \mid t \leq t_s^{sf}(\alpha_q,q)\}$ that is, $t \leq t_s^{sf}(\alpha_{q_{min}}, q_{min})$. As q_M is the chosen quality level, we have also $t \leq t_s^{sf}(\alpha_{q_M}, q_M)$. This condition can be rewritten, by lemma 3:

$$\begin{aligned} & t \leq t_s^{sf}(\alpha_{q_M}, q_M) \\ \Rightarrow \ & t \leq t_s^{sf}(\alpha_{q_M}{}^2, q_{min}) - C^{wc}(\alpha_{q_M}(1), q_M) \\ \Rightarrow \ & t + C^{wc}(a, q_M) \leq t_s^{sf}(\alpha_{q_M}{}^2, q_{min}). \end{aligned}$$

As $C(a,q_M) \leq C^{wc}(a,q_M)$ and $t' = t + C(a,q_M)$, we obtain:

$$t' \leq t_s^{sf}(\alpha_{q_M}{}^2, q_{min}).$$

As $\alpha'_{q_{min}}$ is an EDF schedule of $G/\alpha'_{q_{min}}$, and $\alpha_{q_M}{}^2$ is a schedule of the same precedence graph, by proposition 2 applied to t_s^{sf} and q_{min}, we have $t_s^{sf}(\alpha_{q_M}{}^2, q_{min}) \leq t_s^{sf}(\alpha'_{q_{min}}, q_{min})$, and obtain $t' \leq t_s^{sf}(\alpha'_{q_{min}}, q_{min})$. This demonstrates that, at the next state (α',θ',t'), we have $q_{min} \in \{q \mid t' \leq t_s^{sf}(\alpha'_q, q)\}$.

Thus, for any reachable state (α,θ,t) of $PS(C)||\Gamma$, we have $q_{min} \in \{q \mid t \leq t_s^{sf}(\alpha_q,q)\}$. □

9.3 Propositions 5 and 7

LEMMA 4. *Let α be a sequence of actions and D a deadline function such that D is constant on set(α). Then:*

$$\begin{aligned} t_s^{mx}(\alpha,\theta) &= D - C^{av}(\alpha,\theta) - \delta^{max}(\alpha,\theta) \\ &= t_s^{av}(\alpha,\theta) - \delta^{max}(\alpha,\theta). \end{aligned}$$

Proof of lemma 4: We have:

$$\begin{aligned} t_s^{mx}(\alpha,\theta) &= \mathbf{min}_{1\le k\le|\alpha|}\ t_s^{mx}(\alpha,\theta)(k) \\ &= \mathbf{min}_{1\le k\le|\alpha|}\ D - C^{mx}({}^k\alpha,\theta) \\ &= D - \mathbf{max}_{1\le k\le|\alpha|}\ C^{mx}({}^k\alpha,\theta). \qquad (7) \end{aligned}$$

We rewrite $C^{mx}({}^k\alpha,\theta)$ as $C^{av}({}^k\alpha,\theta) + \delta^{max}({}^k\alpha,\theta)$

$$\begin{aligned} &= C^{av}({}^k\alpha,\theta) + \mathbf{max}_{i\ge1}\ C^{sf}(\alpha[i,k],\theta) - C^{av}(\alpha[i,k],\theta) \\ &= \mathbf{max}_{i\ge1}\ C^{av}(\alpha[1,i-1],\theta) + C^{sf}(\alpha[i,k],\theta). \end{aligned}$$

We conclude that $k_1 \le k_2 \ \Rightarrow\ C^{mx}({}^{k_1}\alpha,\theta) \le C^{mx}({}^{k_2}\alpha,\theta)$. Thus, we rewrite (7) as follows:

$$\begin{aligned} t_s^{mx}(\alpha,\theta) &= D - C^{mx}({}^{|\alpha|}\alpha,\theta) = D - C^{mx}(\alpha,\theta) \\ &= D - C^{av}(\alpha,\theta) - \delta^{max}(\alpha,\theta). \end{aligned}$$

As we have a single deadline, $t_s^{av}(\alpha,\theta) = D - C^{av}(\alpha,\theta)$. We obtain $t_s^{mx}(\alpha,\theta) = t_s^{av}(\alpha,\theta) - \delta^{max}(\alpha,\theta)$. □

LEMMA 5. *Consider $PS(C)||\Gamma$ with the mixed policy $t \le t_s^{mx}$, a function $Best\_Sched^{mx}$ such that it returns schedules that maximize t_s^{mx}. For a single deadline D, if $C = C^{av}$, the schedules (α,θ) computed by $PS(C)||\Gamma$ are such that $i \mapsto \theta(\alpha(i))$ is a non-decreasing function.*

Proof of lemma 5: Let (α,θ,t) and (α',θ',t') be two reachable states of $PS(C)||\Gamma$ such that $(\alpha,\theta,t) \xrightarrow{(a,q)} (\alpha',\theta',t')$. If $\alpha_q = Best\_Sched^{mx}(\alpha,q)$, by lemma 4 we have:

$$\begin{aligned} t &\le t_s^{mx}(\alpha_q,q) = D - C^{av}(\alpha_q,q) - \delta^{max}(\alpha_q,q) \\ t &\le D - C^{av}(\alpha_q(1),q) - C^{av}({\alpha_q}^2,q) - \delta^{max}(\alpha_q,q) \end{aligned}$$

As $C = C^{av}$, $t' = t + C^{av}(a,q) = t + C^{av}(\alpha_q(1),q)$ and

$$t' \le D - C^{av}({\alpha_q}^2,q) - \delta^{max}(\alpha_q,q) \qquad (8)$$

Moreover, since $\delta^{max}(\alpha_q,q) \ge \delta^{max}({\alpha_q}^2,q)$, we obtain from (8) and by lemma 4 $t' \le t_s^{mx}({\alpha_q}^2,q)$.

Let $\alpha'_q = Best\_Sched^{mx}(\alpha',q)$. As α'_q maximizes t_s^{mx}:

$$t' \le t_s^{mx}({\alpha_q}^2,q) \le t_s^{mx}(\alpha'_q,q).$$

This demonstrates that the chosen quality level q at state (α,θ,t) remains possible at the next state (α',θ',t'). □

Proof of proposition 7: The proof is provided for proposition 7 only. By rewriting it without the term δ^{max}, we obtain a proof for proposition 5.

By lemma 5, $i \mapsto \theta(\alpha(i))$ is a non-decreasing function. We know that $\theta(\alpha(1)) = q$. Let i be the last index such that $\theta(\alpha(i)) = q$. Notice that i can be equal to $|\alpha|$. Let t_{i-1} be the actual time after execution of ${}^{i-1}\alpha$ that is, $t_{i-1} = C({}^{i-1}\alpha,\theta) = C({}^{i-1}\alpha,q)$, and $\alpha_{q+1} = Best\_Sched^{mx}({}^{i-1}\alpha,q+1)$ be the planed schedule for the quality level $q+1$. Then, by lemma 4:

$$\begin{aligned} & t_s^{mx}(\alpha_{q+1},q+1) < t_{i-1} \\ \Rightarrow\ & t_s^{av}(\alpha_{q+1},q+1) - \delta^{max}(\alpha_{q+1},q+1) < C({}^{i-1}\alpha,q) \\ \Rightarrow\ & D - \delta^{max}(\alpha,q+1) < C^{av}({}^{i-1}\alpha,q) + C^{av}(\alpha^i,q+1). \end{aligned}$$

where $C^{av}({}^{i-1}\alpha,q) + C^{av}(\alpha^i,q+1)$ satisfies:

$$\begin{aligned} & C^{av}({}^{i-1}\alpha,q) + C^{av}(\alpha^i,q+1) = \\ & C^{av}({}^{i}\alpha,q) + C^{av}(\alpha^{i+1},q+1) + \ C^{av}(a,q+1) - C^{av}(a,q). \end{aligned}$$

For all $j \le i$, $\theta(\alpha(i)) = q$. If $i < |\alpha|$, for all $j > i$, by lemma 5, $\theta(\alpha(i)) \ge q+1$. Thus, $C^{av}({}^{i}\alpha,q) + C^{av}(\alpha^{i+1},q+1) \le C^{av}(\alpha,\theta)$. Moreover, by definition of Δ, we have $C^{av}(a,q+1) - C^{av}(a,q) \le \Delta$. We obtain:

$$D - \delta^{max}(\alpha,q+1) - \Delta < C(\alpha,\theta). \quad \square$$

9.4 Proposition 8

Proof of proposition 8:

• The deadline function D^* is constant on G/A_l. Thus, by lemma 4 we have, for any schedule α_l of G/A_l, $t_s^{mx}(\alpha_l,q) = t_s^{av}(\alpha_l,q) - \delta^{max}(\alpha_l,q)$, where $t_s^{av}(\alpha_l,q) = D^*(\alpha_l) - C^{av}(\alpha_l,l)$ do not depend on α_l. We obtain:

$$\begin{aligned} & \mathbf{max}\ \{\ t_s^{mx}(\alpha_l,q) \mid \alpha_l \in Sched(G/A_l)\ \} \\ = \ & t_s^{av}(\alpha_l,q) - \mathbf{min}\ \{\ \delta^{max}(\alpha_l,q) \mid \alpha_l \in Sched(G/A_l)\ \}. \end{aligned}$$

• Let $A_1 \dots A_l$ be the partition induced by D^*, $\alpha = \alpha_1 \dots \alpha_L$ and $\alpha' = \alpha'_1 \dots \alpha'_L$ be two EDF schedules such that, for all $l \in \{1,\dots,L\}$, α_l and α'_l are schedules of G/A_l. Assume that, for all l, α_l is an EDF-optimal schedule that is, $\delta^{max}(\alpha_l,q) \le \delta^{max}(\alpha'_l,q)$.

Let k_l be the length of the schedule $\alpha_1 \dots \alpha_l$. We have $t_s^{mx}(\alpha,q) = \mathbf{min}\ \{\ t_s^{mx}(\alpha,q)(k_l)\ \}$, and:

$$\begin{aligned} & t_s^{mx}(\alpha,q)(k_l) = t_s^{av}(\alpha,q)(k_l) - \delta^{max}(\alpha_1 \dots \alpha_l,q) \\ = \ & t_s^{av}(\alpha,q)(k_l) - \mathbf{max}\ \{\ \delta^{max}(\alpha_j,q) + \beta(\alpha_{j+1} \dots \alpha_l,q) \mid \\ & \qquad 1 \le j \le l\ \}. \end{aligned}$$

Moreover since $\beta(\alpha_{i+1} \dots \alpha_l,q) = \beta(\alpha'_{i+1} \dots \alpha'_l,q)$ and $\delta^{max}(\alpha_j,q) \le \delta^{max}(\alpha'_j,q)$ for all j, we have:

$$\begin{aligned} & \mathbf{max}\ \{\ \delta^{max}(\alpha_j,q) + \beta(\alpha_{j+1} \dots \alpha_l,q) \mid 1 \le j \le l\ \} \\ \le \ & \mathbf{max}\ \{\ \delta^{max}(\alpha'_j,q) + \beta(\alpha'_{j+1} \dots \alpha'_l,q) \mid 1 \le j \le l\ \}. \end{aligned}$$

As $t_s^{av}(\alpha,q)(k_l) = t_s^{av}(\alpha',q)(k_l)$ we have, for all l, $t_s^{mx}(\alpha,q)(k_l) \le t_s^{mx}(\alpha',q)(k_l)$. We conclude that $t_s^{mx}(\alpha,q) \le t_s^{mx}(\alpha',q)$. □

9.5 Proposition 9

Proof of proposition 9: Let q be a quality level. Then, $\delta(\alpha'^j,q) = \delta(\alpha^j,q)$ for $j \notin \{i,i+1\}$. In the following, we denote by $X(\alpha)$ the $X(\alpha,q)$, where X can be δ, η or β.

• **R1:** Suppose that a and b are such that $\beta(a) \le 0$, $\beta(b) \le 0$ and $\eta(a) < \eta(b)$. It is obvious that $\delta(\alpha^{i+1}) > \delta(\alpha^i)$. We obtain $\delta(\alpha'^i) \le \delta(\alpha^{i+1})$ and $\delta(\alpha'^{i+1} \le \delta(\alpha^{i+1})$. We conclude that $\delta^{max}(\alpha',q) \le \delta^{max}(\alpha,q)$.

• **R2:** Suppose that a and b are such that $\beta(a) \le 0$, $\beta(b) > 0$. Then $\delta(\alpha'^i) \le \delta(\alpha^{i+1})$ and $\delta(\alpha'^{i+1}) \le \delta(\alpha^i)$. We conclude that $\delta^{max}(\alpha',q) \le \delta^{max}(\alpha,q)$.

• **R3:** Suppose that a and b are such that $\beta(a) > 0$, $\beta(b) > 0$, and $(\eta - \beta)(a) > (\eta - \beta)(b)$. It can be shown that $\delta(\alpha^i) > \delta(\alpha^{i+1})$. It is straightforward that we have $\delta(\alpha'^i) \le \delta(\alpha^i)$ and $\delta(\alpha'^{i+1}) \le \delta(\alpha^i)$. We conclude that $\delta^{max}(\alpha',q) \le \delta^{max}(\alpha,q)$. □