Proc. of the 2nd Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS 2009), Washington, D.C., USA, Dec. 1, 2009.

Selecting the minimum consumed bandwidth of an EDF task set*

Enrico Bini, Giorgio Buttazzo, Yifan Wu
Scuola Superiore Sant'Anna, Pisa, Italy
{e.bini,g.buttazzo,y.wu}@sssup.it

Abstract

The usage of virtual processor is a key aspect for isolating real-time applications. A typical interface of virtual processor is expressed by the bandwidth and the delay.

After formulating the overall consumed bandwidth as a function of the bandwidth and the delay that accounts also for the context switches, we proposed an algorithm that find the least consuming interface that is capable to guarantee the deadlines of a set of tasks scheduled by EDF. The proposed algorithm is inspired by the dual simplex algorithm.

1 Introduction

The advancement of computer architectures allows to execute more and more applications on the same physical platform. When executing many real-time applications in the same system, it is important to compose them in a way that a misbehavior occurring in one application does not affect the others. The environment allowing the composition of application has in the past been called *open environment* [7]. These environments abstract the computing resource provided by the physical processor, by a *virtual processor* (VP) that provides only a fraction of the available resource. This is implemented by introducing two schedulers: a *global scheduler* that assign the physical processor to the virtual ones, and a *local scheduler* that in turn selects the tasks of the application to be run on the VP. This problem is often also called *hierarchical scheduling* because it allows the composition of different scheduling policies at many levels.

VPs are characterized by *interfaces* that describe the amount of computation provided by the VP to the application. In this way it is possible to guarantee the real-time properties of the application based on the only interface of the VP, unbinding the analysis of an application from the others.

*This work has been partially supported by the ACTORS European project under contract 216586.

In the context of hierarchical scheduling, the *feasibility problem* consists in determining whether or not a given application can meet all the real-time constraints (deadlines) when running onto a VP characterized by a given interface. Although the feasibility problem sets the foundation of future research, it assumes that the VP is known a priori. Instead, the *design problem* attempts to respond properly to the need of the application designer: what is the "best" virtual processor that allows the application to meet all the deadlines? Usually the quality of a VP is measured by the amount physical resource it consumes.

Contribution of the paper In this paper, first we formalize our notion of optimality of a virtual processor. Then we derive the optimal interface of an application modeled by a set of tasks scheduled by EDF.

1.1 Related works

The virtualization of computing resource is a constantly active research area. Mercer et al. [14] proposed a resource reservation mechanism based on a required budget and period to provide an abstraction of a uniprocessor with reduced speed. Deng and Liu [7] proposed a two-level hierarchical architecture, which uses the EDF as global scheduler and a dedicated Total Bandwidth Server (TBS) [20] for each application. The paper presents also sufficient conditions for schedulability. This work has been later extended by Kuo et al. [10] for using RM as global scheduling algorithm, however the authors assume that all tasks are periodic with harmonic periods. Abeni and Buttazzo [1] proposed the Constant Bandwidth Server (CBS) to isolate an application requiring a varying amount of computation on a virtual processor with reduced speed. Lipari and Baruah in [12] presented the Bandwidth Sharing Server (BSS) scheduling algorithm that uses EDF as global scheduling algorithm, and permits to select any scheduling algorithm as application level scheduler. The paper presents schedulability conditions for applications that use EDF and RM as second level schedulers. However, the algorithm is complex to implement and assumes the knowledge of all task deadlines.

Saewong et al. [16] proposed to use the Deferrable Server in a hierarchical way. They present a schedulability analysis that is based on the worst-case response time for a local fixed priority scheduler. Davis and Burns [6] suggested that binding the server period with the period of the tasks leads to better resource usage.

Mok, Feng, and Chen [15] introduced the concept of "supply function" of a static time partition to measure the minimum amount of computing resource provided. Later, Almeida and Pedreiras [2] applied similar techniques to schedule messages over the FTT-CAN network. Lipari and Bini [13] derived the set of supply functions that can feasibly schedule a given application. Shin and Lee [19] introduced the periodic resource model (that is a special class of supply functions) also deriving a utilization bound, later extended by Easwaran et al. [8] to account for a server deadline possibly different than the period. Wandler and Thiele [21] proposed the concept of interface-based design which uses network calculus [11] to compute the supply functions of each component.

Shin et al. [18] investigated the selection of the optimal interface also accounting for resources shared among different applications. However, the period was assumed to be given, while in this paper we also explore the possible values of period (releasing indeed the hypothesis of shared resources).

Very recently, Fisher and Dewan [9] proposed both a fully polynomial time approximation scheme (FPTAS) to compute the minimum budget Q of a VP implemented by a periodic server with a given period P and deadline D. However the application designer still has to determine the period and the deadline of the server.

2 System model

To ease the readability throughout the paper we use the notation $(\cdot)_0$ as a shortcut for $\max\{0, \cdot\}$.

2.1 Application model

An application $\mathcal{T}$ is modeled by a set of n tasks $\{\tau_1, \ldots, \tau_n\}$. Each task τ_i has a computation time C_i, a period T_i, and a deadline D_i. We also introduce the task utilization $U_i = \frac{C_i}{T_i}$. All these numbers are positive integers.

The computational requirement of the application $\mathcal{T}$ is modeled by its *demand bound function*, that is:

$$\mathsf{dbf}(t) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \left(\left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right)_0 C_i \qquad (1)$$

A necessary and sufficient schedulability test for EDF consists in checking that the demand never exceeds the length of the interval on every processor [3]:

$$\forall t > 0 \quad \mathsf{dbf}(t) \leq t \qquad (2)$$

The demand bound function $\mathsf{dbf}(t)$ is the sum of floor functions. Hence it is right-continuous, piecewise constant, and increasing. It follows that it can be represented by the values at each step. For this purpose we introduce the set of *scheduling points* $\mathcal{P}$:

$$\mathcal{P} \stackrel{\text{def}}{=} \{(t; w) \,:\, \mathsf{dbf}(t) = w,\ \lim_{x \to t^-} \mathsf{dbf}(x) < w\} \qquad (3)$$

The usefulness of the set of scheduling points $\mathcal{P}$ is demonstrated by the following (indeed very simple) lemma.

Lemma 1 *Let* dbf *and* $\mathcal{P}$ *be the demand bound function of a task set and the corresponding set of scheduling points. Then for any non decreasing function* $f : \mathbb{R} \to \mathbb{R}$*, we have*

$$\forall t > 0 \quad \mathsf{dbf}(t) \leq f(t) \qquad (4)$$

if and only if

$$\forall (t, w) \in \mathcal{P} \quad w \leq f(t) \qquad (5)$$

Proof The implication (4)⇒(5) is trivial. In fact, by definition (Eq. (3))

$$w = \mathsf{dbf}(t) \leq f(t).$$

For proving (5)⇒(4), let t be any point and $t^* = \sup\{x \leq t : (x, w) \in \mathcal{P}\}$. Since the dbf is constant in $[t^*, t]$, we have

$$\mathsf{dbf}(t) = \mathsf{dbf}(t^*) \leq f(t^*) \leq f(t)$$

since f is non-decreasing, which concludes the proof. □

2.2 The interface of the virtual processor

The interface of the virtual processor is an abstraction of the fraction of computing resource supplied by the physical processor. One of the key concept in the model of the computing resource is the supply function, that is recalled below.

Definition 1 (Def. 9 in [15], Th. 1 in [13], Eq. (6) in [8]) *The* supply function $Z(t)$ *of a virtual processor is the minimum amount of time provided in every time interval of length* $t \geq 0$*, by the global scheduler to the application.*

For example, for the simple case of a periodic server that allocates Q units of time every period P [13, 19], the supply function is:

$$Z(t) = \max\{0, t - P + Q - (k+1)(P - Q), kQ\} \qquad (6)$$

with $k = \left\lfloor \frac{t-P+Q}{P} \right\rfloor$.

Mok et al. [15] introduced the "bounded delay partition" to describe a VP by two parameters only: a bandwidth α and a delay Δ. The bandwidth α measures the amount of resource that is assigned to the demanding application, whereas Δ represents the worst-case service delay.

Given the supply function Z, the bandwidth α and the delay Δ can be formally defined as follows.

Definition 2 (compare Def. 5 in [15]) *Given a virtual processor with supply function Z, the* bandwidth α *of the virtual processor is defined as*

$$\alpha \stackrel{\text{def}}{=} \lim_{t \to \infty} \frac{Z(t)}{t}. \tag{7}$$

The Δ parameter provides a measure of the responsiveness, as proposed by Mok et al. [15].

Definition 3 (compare Def. 14 in [15]) *Given a virtual processor with supply function Z and bandwidth α, the* delay Δ *of the virtual processor is defined as*

$$\Delta \stackrel{\text{def}}{=} \sup_{t \geq 0} \left\{ t - \frac{Z(t)}{\alpha} \right\}. \tag{8}$$

Informally speaking, given a VP with bandwidth α, the delay Δ is the minimum horizontal displacement such that the line $\alpha(t-\Delta)$ is a lower bound of $Z(t)$. Hence, the supply function Z of a VP can be lower bounded as follows:

$$Z(t) \geq \alpha(t-\Delta)_0. \tag{9}$$

If the VP is implemented through a periodic server [13, 19] that allocates a budget Q every period P, then the bandwidth and delay are:

$$\alpha = \frac{Q}{P} \qquad \Delta = 2(P-Q). \tag{10}$$

In practice, however, a portion of the processor bandwidth is wasted to perform context switches every time a server is executed. If σ is the runtime overhead required for a context switch, and P is the server period, the consumed server bandwidth can be computed as:

$$B = \frac{Q+\sigma}{P} = \alpha + \frac{\sigma}{P}.$$

which can be expressed as a function of α and Δ as follows

$$B = \alpha + 2\sigma \frac{1-\alpha}{\Delta}. \tag{11}$$

Hence we target the minimization of the consumed bandwidth, as expressed in Eq. (11), that provides enough computing resource to meet the deadlines of the tasks in $\mathcal{T}$.

3 Computing the minimum bandwidth

Since we abstract the VP by the bandwidth α and delay Δ, we know [19] that the task set is schedulable over the VP if:

$$\forall t \geq 0 \quad \mathsf{dbf}(t) \leq \alpha(t-\Delta)_0, \tag{12}$$

which can be restated (thanks to Lemma 1) also as

$$\forall (t;w) \in \mathcal{P} \quad w \leq \alpha(t-\Delta)_0. \tag{13}$$

Now the problem is to select the (α, Δ) parameters among all possible pairs that satisfy Eq. (13). We choose to select the pair that minimizes the bandwidth B used by the virtual processor, as given by Eq. (11), since this bandwidth accounts both for the active bandwidth α and the bandwidth wasted due to context switches. Hence, the best (α,Δ) pair is the solution of the following minimization problem:

$$\begin{aligned} &\text{minimize} \quad \alpha + \varepsilon \frac{1-\alpha}{\Delta} \\ &\text{subject to} \quad w \leq \alpha(t-\Delta)_0, \quad \forall (t;w) \in \mathcal{P} \end{aligned} \tag{14}$$

with $\varepsilon = 2\sigma$.

Such a minimization problem can be solved very efficiently, thanks to the good properties of both the constraint and the cost function. We first prove the convexity of the constraint.

Lemma 2 *Given $t, w > 0$, let $D(t,w)$ be defined as*

$$D(t,w) = \{(\alpha,\Delta) \in \mathbb{R}^2 : \alpha(t-\Delta) \geq w, \alpha \geq 0\} \tag{15}$$

then $D(t,w)$ is convex.

Proof We start observing that

$$\alpha(t-\Delta) \geq w \geq 0 \;\Rightarrow\; t - \Delta \geq 0 \tag{16}$$

because $\alpha \geq 0$. To prove the convexity of $D(t,w)$ we use the property that

$$\{(x,y) : f(x) \leq y\} \text{ is convex} \Leftrightarrow \frac{\mathrm{d}^2 f}{\mathrm{d}x^2} \geq 0 \tag{17}$$

In fact we have

$$D(t,w) = \left\{ \frac{w}{t-\Delta} \leq \alpha \right\}$$

Now

$$\frac{\mathrm{d}^2}{\mathrm{d}\Delta^2} \frac{w}{t-\Delta} = \frac{2w}{(t-\Delta)^3} \geq 0$$

because of Eq. (16). Hence from the property of Eq. (17), the Lemma follows. □

Figure 1 shows examples of the domains $D(t,w)$.

Regarding the properties of the cost function, we first recall the following definition.

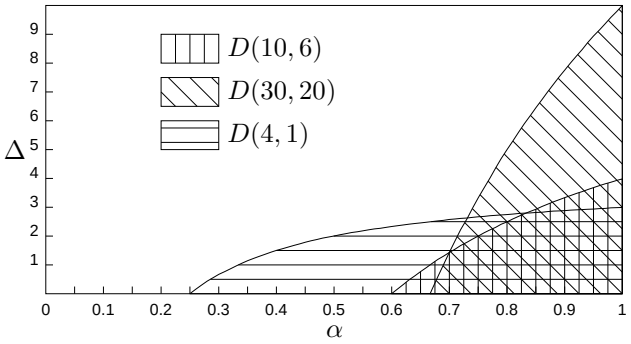

Figure 1. Examples of the regions $D(t,w)$.

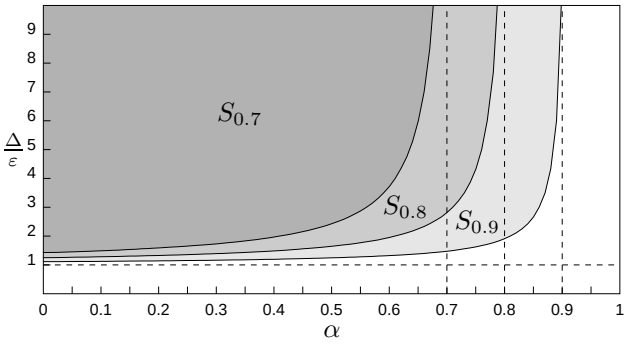

Figure 2. Examples of the regions S_B.

Definition 4 (Section 3.4 in [5]) *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called* quasiconvex *if its domain and all its sublevel sets $S_B = \{x \in \mathbf{dom} f : f(x) \leq B\}$, for $v \in \mathbb{R}$, are convex.*

Notice that convexity implies quasiconvexity, but the viceversa is not true [5]. We have the following result.

Lemma 3 *The function $g : [0,1] \times (0,+\infty) \to \mathbb{R}$*

$$g(\alpha, \Delta) = \alpha + \varepsilon \frac{1-\alpha}{\Delta}$$

is quasiconvex.

Proof We first notice that the domain of g, that is $G = [0,1] \times (0,+\infty)$ is convex. From the definition of quasiconvexity we have to prove that all the level sets

$$S_B = \left\{ (\alpha, \Delta) \in G : \alpha + \varepsilon \frac{1-\alpha}{\Delta} \leq B \right\} \qquad (18)$$

are convex (see Figure 2 for graphical representation). Since B is interpreted as the overall bandwidth used by the reservation, we only need to prove this for $B \leq 1$. Since $B \geq \alpha$ and $\Delta \geq 0$, we have that:

$$\alpha + \varepsilon \frac{1-\alpha}{\Delta} \leq B \quad \Leftrightarrow \quad \varepsilon \frac{1-\alpha}{B-\alpha} \leq \Delta$$

and from the property of Eq. (17),

$$\left\{ (\alpha, \Delta) : \varepsilon \frac{1-\alpha}{B-\alpha} \leq \Delta \right\} \Leftrightarrow \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \frac{1-\alpha}{B-\alpha} \geq 0$$

since $k > 0$.

We have

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} \frac{1-\alpha}{B-\alpha} = \frac{-1(B-\alpha) + (1-\alpha)}{(B-\alpha)^2} = \frac{1-v}{(B-\alpha)^2}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \frac{1-\alpha}{B-\alpha} = 2\frac{1-B}{(B-\alpha)^3}$$

that is greater than or equal to zero, because $B \leq 1$ and $\alpha \leq B$. This proves the convexity of the level sets S_B and the quasiconvexity of g as required. □

Since the cost function of the problem of Eq. (14) is quasiconvex (see Lemma 3) and the feasibility region is the intersection of convex regions (from Lemma 2), then the minimization problem is a standard quasiconvex optimization problem [5], which can be solved very efficiently by standard techniques.

3.1 The optimization algorithm

Our proposed solution is an adaptation of the dual simplex algorithm to convex problems.

For convenience we enumerate the scheduling points in $\mathcal{P}$ to be able to rewrite the problem of Eq. (14) as follows:

$$\begin{aligned} &\text{minimize} \quad \alpha + \varepsilon \frac{1-\alpha}{\Delta} \\ &\text{subject to} \quad w_i \leq \alpha(t_i - \Delta)_0, \quad \forall i = 1, \ldots, m \end{aligned} \qquad (19)$$

We distinguish between several possibilities.

Case 1: $\varepsilon = 0$ If ε, that is a coefficient that accounts for the context switch overhead, is equal to zero (in the ideal case), then the minimization problem is greatly simplified. It becomes

$$\begin{aligned} &\text{minimize } \alpha \\ &\text{subject to } \alpha(t_i - \Delta) \geq w_i, \quad \forall i \end{aligned}$$

$$\forall i \; \alpha \geq \frac{w_i}{t_i - \Delta} \quad \Rightarrow \quad \alpha \geq \sup_i \frac{w_i}{t_i - \Delta}$$

from which it follows

$$\Delta = 0, \quad \alpha = \sup_i \frac{w_i}{t_i} \qquad (20)$$

that is the solution in absence of context switch overhead. A similar result was found in the context of power-aware scheduling [17, 4].

Case 2: $\varepsilon > 0$ If it exists some cost for switching from one VP to another, then we must set $\varepsilon > 0$. In this case, we aim at solving the problem of Eq. (19) through the Lagrange multipliers method. The Lagrange function is:

$$L = \alpha + \varepsilon\frac{1-\alpha}{\Delta} + \sum_i \mu_i(w_i - \alpha(t_i - \Delta)) \qquad (21)$$

The partial derivatives are:

$$\frac{\partial L}{\partial \alpha} = 1 - \frac{\varepsilon}{\Delta} - \sum_i \mu_i(t_i - \Delta) \qquad (22)$$

$$\frac{\partial L}{\partial \Delta} = -\varepsilon\frac{1-\alpha}{\Delta^2} + \alpha\sum_i \mu_i \qquad (23)$$

To search for the minimum we evaluate all the possible stationary points. Below we say that a constraint is adherent to a point when the corresponding inequality holds with the equal sign at that point. Notice that, since the optimization is developed on the bi-dimensional space of the server parameters α and Δ, if there are more than two constraints adherent to a point, then the same solution can be found from any two constraints among the adherent ones.

Case 2A: one adherent constraint Let i be the index of the adhering constraint. In this case, the boundary of $D(t_i, w_i)$ must be tangent to the boundary of S_B, and the tangent point is our stationary point. See Figure 3 for a graphical representation.

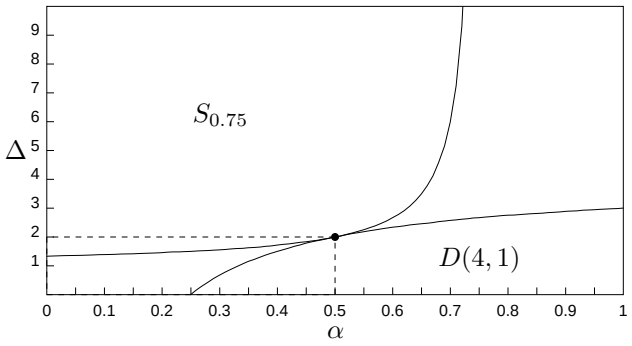

Figure 3. One adherent constraint.

From the condition that the i constraint is adherent, it follows that $\mu_j = 0,\ \forall j \neq i$. Hence, the equations that needs to be verified are

$$1 - \frac{\varepsilon}{\Delta} - \mu_i(t_i - \Delta) = 0 \qquad (24)$$

$$-\varepsilon\frac{1-\alpha}{\Delta^2} + \alpha\,\mu_i = 0 \qquad (25)$$

$$\alpha(t_i - \Delta) = w_i \qquad (26)$$

From these three equations we can derive a second degree polynomial which can be solved directly. In fact, it follows

$$\begin{aligned}
&\alpha(\Delta^2 - \varepsilon\Delta) - \varepsilon(1-\alpha)(t_i - \Delta) = 0\\
&\alpha = \frac{w_i}{t_i - \Delta}, \quad 1 - \alpha = \frac{t_i - \Delta - w_i}{t_i - \Delta}\\
&w_i(\Delta^2 - \varepsilon\Delta) - \varepsilon(t_i - \Delta - w_i)(t_i - \Delta) = 0\\
&(w_i - \varepsilon)\Delta^2 + 2\varepsilon(t_i - w_i)\Delta - \varepsilon t_i(t_i - w_i) = 0 \qquad (27)
\end{aligned}$$

Then if we set

$$a = \frac{w_i - \varepsilon}{\varepsilon(t_i - w_i)} \qquad (28)$$

from Eq. (27) it follows

$$\begin{aligned}
&a\Delta^2 + 2\Delta - t_i = 0\\
&\Delta = \frac{\sqrt{1 + at_i} - 1}{a} \qquad (29)
\end{aligned}$$

If we assume, as in Figure 3, $t_i = 4, w_i = 1, \varepsilon = 1$ the solution found is $\alpha = 0.5, \Delta = 2$ and corresponding consumed bandwidth is $B = 0.75$.

Case 2B: two adherent constrains Let i and j be the indexes of the two adhering constraints. In this case we have $\mu_k = 0,\ \forall k \neq i, k \neq j$. The equations that needs to be verified are

$$\alpha(t_i - \Delta) = w_i \qquad (30)$$

$$\alpha(t_j - \Delta) = w_j \qquad (31)$$

$$\frac{w_i}{t_i - \Delta} = \frac{w_j}{t_j - \Delta} \qquad (32)$$

$$w_i(t_j - \Delta) = w_j(t_i - \Delta) \qquad (33)$$

$$\Delta = \frac{w_i t_j - w_j t_i}{w_i - w_j} \qquad (34)$$

from which it follows that

$$\alpha = \frac{w_i - w_j}{t_i - w_j} \qquad (35)$$

Selection of the constraints In the previous two cases we found the stationary points for a given constraint i (or a given pair (i, j) of constraints). However we didn't explain how to select the constraints. Now we conclude the description of the method by explaining this selection procedure.

The proposed technique is inspired by the dual simplex algorithm. It consists in updating a basic dual feasible solution (BDFS) until it is also feasible. However, differently than in the simplex algorithm, a BDFS can be constituted also by only one constraint index (while in the classic dual simplex algorithm a solution is always composed by two linearly independent constraints).

The algorithm is the following.

1. We initialize $\mathsf{BDFS} = \{1\}$ (the first constraint), and we compute the current solution (α^*, Δ^*) by using the solution of case 2A, since BDFS has only one constraint index.

2. If the solution (α^*, Δ^*) satisfies all the constraints then it is both dual and primal feasible, hence it is the optimum and we can exit the algorithm. Otherwise, let k be the first constraint index that is not satisfied (k is the *entering index*).

3. We evaluate the stationary points corresponding to $\{k\}$ and $\{\{k, x\} : x \in \mathsf{BDSF}\}$. This point is computed according to case 2A (or 2B) if the cardinality of BDFS is one (or two, respectively). We set the current solution (α^*, Δ^*) equal to the one that has a higher cost among the computed ones, and we set BDFS equal to the set that generated it. This selection is necessary to keep also the next solution dual feasible. We jump to step 2.

The termination of the algorithm is guaranteed by the finiteness of the number of scheduling points in $\mathcal{P}$ and by the Bland's anti-cycling rule at step 2 that prevents to return to a previously visited BDFS.

The complexity of the algorithm is the same as the number of scheduling points.

4 Experiments

In this last section we show the results of the proposed optimization method for a simple task set whose parameters are shown in Table 1. We highlight that the total utilization

i	C_i	T_i	D_i
1	1	8	6
2	2	10	13
3	3	16	15

Table 1. Example of task set.

of this task set is $U = 0.5125$.

We solved the problem of the optimal bandwidth selection for varying values of the overhead ε. The results are reported in the next figures.

Figure 4 shows both the optimal server bandwidth α (by a solid line) and the consumed bandwidth that accounts also for the overhead cost (dashed line). The simulation is run only for values of the overhead ε that make the overall bandwidth not larger 1.

For the same experiments, Figure 5 shows the achieved optimal values of the delay Δ (solid gray line). For convenience, we also show the server period P (solid black line) and budget Q (dashed line), assuming a strictly periodic server, that is

$$P = \frac{\Delta}{1-\alpha} \qquad Q = \frac{\alpha\Delta}{1-\alpha}$$

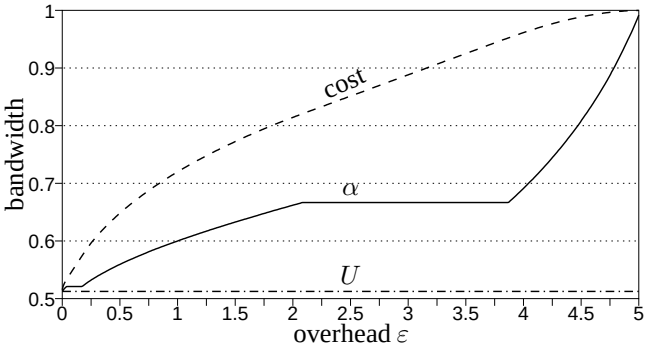

Figure 4. Values of P, Q, Δ.

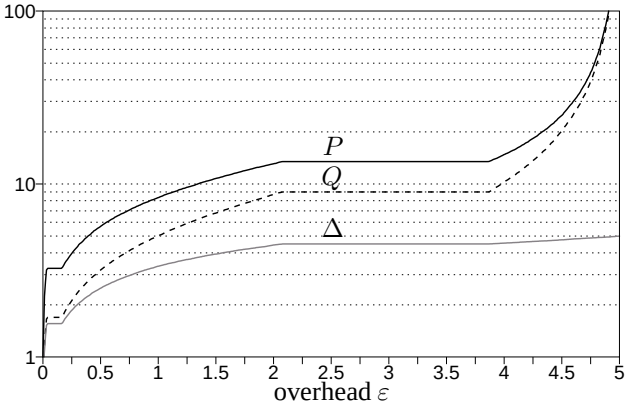

Figure 5. Values of P, Q, Δ.

5 Conclusions

We investigated the optimal interface selection of a virtual processor that is abstracted by the only bandwidth α and delay Δ. We showed several interesting properties of the formulated optimization problem (such as quasiconvexity) that allow an efficient solution.

References

[1] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 4–13, Madrid, Spain, December 1998.

[2] Luís Almeida, Paulo Pedreiras, and José Alberto G. Fonseca. The FTT-CAN protocol: Why and

how. *IEEE Transaction on Industrial Electronics*, 49(6):1189–1201, December 2002.

[3] Sanjoy K. Baruah, Rodney Howell, and Louis Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.

[4] Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari. Speed modulation in energy-aware real-time systems. In *Proceedings of the* 17^{th} *Euromicro Conference on Real-Time Systems*, pages 3–10, Palma de Mallorca, Spain, July 2005.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. available at `http://www.stanford.edu/~boyd/cvxbook.html`.

[6] Robert Davis and Alan Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proceedings of the* 26^{th} *IEEE International Real-Time Systems Symposium*, pages 389–398, Miami (FL), U.S.A., December 2005.

[7] Zhong Deng and Jane win-shih Liu. Scheduling real-time applications in Open environment. In *Proceedings of the* 18^{th} *IEEE Real-Time Systems Symposium*, pages 308–319, San Francisco, CA, U.S.A., December 1997.

[8] Arvind Easwaran, Madhukar Anand, and Insup Lee. Compositional analysis framework using EDP resource models. In *Proceedings of the* 28^{th} *IEEE International Real-Time Systems Symposium*, pages 129–138, Tucson, AZ, USA, 2007.

[9] Nathan Fisher and Farhana Dewan. Approximate bandwidth allocation for compositional real-time systems. In *Proceedings of the* 21^{st} *Euromicro Conference on Real-Time Systems*, pages 87–96, Dublin, Ireland, July 2009.

[10] Tei-Wei Kuo and Ching-Hui Li. Fixed-priority-driven open environment for real-time applications. In *Proceedings of the* 20^{th} *IEEE Real-Time Systems Symposium*, pages 256–267, Phoenix, AZ, U.S.A., December 1999.

[11] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus*, volume 2050 of *Lecture Notes in Computer Science*. Springer, 2001.

[12] Giuseppe Lipari and Sanjoy K. Baruah. Efficient scheduling of multi-task applications in open systems. In *Proceedings of the* 6^{th} *Real-Time Systems and Applications Symposium,*, pages 166–175, Washington, DC U.S.A., June 2000.

[13] Giuseppe Lipari and Enrico Bini. Resource partitioning among real-time applications. In *Proceedings of the* 15^{th} *Euromicro Conference on Real-Time Systems*, pages 151–158, Porto, Portugal, July 2003.

[14] Clifford W. Mercer, Stefan Savage, and Hydeyuki Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 90–99, Boston, MA, U.S.A., May 1994.

[15] Aloysius K. Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Proceedings of the* 7^{th} *IEEE Real-Time Technology and Applications Symposium*, pages 75–84, Taipei, Taiwan, May 2001.

[16] Saowanee Saewong, Ragunathan Rajkumar, John P. Lehoczky, and Mark H. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proceedings of the* 14^{th} *Euromicro Conference on Real-Time Systems*, pages 152–160, Wien, Austria, June 2002.

[17] Kiran Seth, Aravindh Anantaraman, Frank Mueller, and Eric Rotenberg. FAST: Frequency-aware static timing analysis. In *Proceedings of the* 24^{th} *IEEE Real-Time Systems Symposium*, pages 40–51, Cancun, Mexico, December 2003.

[18] Insik Shin, Moris Behnam, Thomas Nolte, and Mikael Nolin. Synthesis of optimal interfaces for hierarchical scheduling with resourcesq. In *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, pages 209–220, Barcelona, Spain, December 2008.

[19] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, pages 2–13, Cancun, Mexico, December 2003.

[20] Marco Spuri and Giorgio C. Buttazzo. Scheduling aperiodic tasks in dynamic priority systems. *Journal of Real-Time Systems*, 10(2):179–210, March 1996.

[21] Ernesto Wandeler and Lothar Thiele. Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling. In *Proceedings of the* 5^{th} *International Conference on Embedded Software*, pages 80–89, Jersey City (NJ), USA, September 2005.