

Distributed Management of CPU Resources for Time-Sensitive Applications

Georgios Chasparis, Martina Maggio, Karl-Erik Årzén, Enrico Bini
Lund University, Sweden

Abstract—The number of applications sharing the same embedded device is increasing dramatically. Very efficient mechanisms (resource managers) for assigning the CPU time to all demanding applications are needed. Unfortunately, existing optimization-based resource managers consume too much resource themselves. In this paper, we address the problem of distributed convergence to fair allocation of CPU resources for time-sensitive applications. We propose a novel resource management framework where both applications and the resource manager act independently trying to maximize their own performance measure according to a utility-based adjustment process. Contrary to prior work on centralized optimization schemes, the proposed framework exhibits adaptivity and robustness to changes both in the number and nature of applications, while it assumes minimum information available to both applications and the resource manager. It is shown analytically that fair resource allocation can be achieved through the proposed adjustment process when the CPU is overloaded. Experiments using the TrueTime Matlab toolbox show the validity of the proposed approach.

I. INTRODUCTION

A current trend in embedded computing is that the number of applications that should share the same execution platform increases. This is due to the increase in capacity of new hardware platforms, e.g., through the use of multi-core techniques, as well as the increase of user demands. An example includes the move from federated to integrated system architectures in the automotive industry.

As the number of applications increases, the need for better mechanisms for controlling the execution behavior of the applications becomes apparent. Increasingly often, virtualization or resource reservation techniques [1], [2] are used. According to these techniques, each reservation is viewed as a virtual processor executing at a fraction of the speed of the physical processor, i.e., the *bandwidth* of the reservation, while the tasks in the different reservations are temporally isolated from each other. Another trend in embedded computing is an increase in temporal uncertainty, both due to the increased hardware complexity, e.g., shared cache hierarchies, and the increased chip density. Hence, using dynamic adaptation is crucial. In the resource reservation case this means that the bandwidth assignment is decided on-line based on feedback from the applications.

An orthogonal dimension along which the performance of an application can be tuned is the selection of its *service level*. It is assumed that an application is able to execute

at different service levels, where a higher service level implies a higher quality-of-service (QoS) at the price of higher resource consumption. Examples are adjustable video resolutions, the amount of data sent through a socket channel to render a web page, and the possibility to execute a controller at different sampling rates.

The typical solution to this problem is the implementation of a *resource manager* (RM), which is in charge of:

- assigning the resources to each application;
- monitoring the use of resources and possibly adjusting the assignment based on measurements;
- assigning the service levels to each application, so that an overall delivered quality is maximized.

This is often done through the use of *optimization* and *feedback* from the application.

Resource managers that are based on the concept of feedback, monitor the progress of the applications and adjust the resource management based on measurements [3], [4]. In these early approaches, however, quality adjustment was not considered. Cucinotta et al. [5] proposed an inner loop to control the resource allocation nested within an outer loop that controls the overall delivered quality.

Optimization-based resource managers have also received a considerable attention [6], [7]. These approaches, however, rely on the solution of a centralized optimization problem that determines *both* the amount of the assigned resources and the service levels of all applications [6], [8], [9]. In the context of networking, Johansson et al. [10] modeled the service provided by a set of servers to workloads belonging to different classes as a utility maximization problem. However, there is no notion of adjustment of the service level of the applications.

An example of combined use of optimization and feedback was developed in the ACTORS project [9], [11]. In that project, applications provide a table to the RM describing the required amount of CPU resources and the expected QoS achieved at each supported service level [9], [11]. In the multi-core case, applications are partitioned over the cores and the amount of resources is given for each individual partition. Then, the RM decides the service level of all applications and how the partitions should be mapped to physical cores using a combination of integer linear programming (ILP) and first-fit bin-packing.

On-line centralized optimization schemes have several weaknesses. First, the complexity of the solvers used to implement the RM (such as ILP solvers) grows significantly with the number of applications. Hence, it is impractical to have a RM that optimally assigns resources at the price of

This work was supported by the Linneaus Center LCCC, the Swedish VR project n.2011-3635 “Feedback-based resource management for embedded multicore platform”, and the Marie Curie Intra European Fellowship within the 7th European Community Framework Programme.

a large consumption of resources by the RM itself. Second, to enable a meaningful formulation of a cost function in such optimization problems, the RM must compare the quality delivered by different applications. This comparison is unnatural because the concept of quality is extremely application dependent. Finally, solving for both the amount of assigned resources and the service levels for all applications, as in most centralized schemes [6], [8], [9], could be computationally demanding. Besides, a proper assignment of service levels requires application knowledge. In particular, the application must inform the RM about its available service levels and the expected consumed resources at each service level. Thus, service levels would better be assigned by the applications rather than the RM.

To this end, distributed optimization schemes have recently attracted considerable attention. Subrata et al. [12] considered a cooperative game formulation for job allocation to several service providers in grid computing. Job arrivals follow a Poisson process and a centralized optimization problem is formulated for computing Nash bargaining solutions. Wei et al. [13] proposed a non-cooperative game-theoretic formulation to allocate computational resources to a given number of tasks in cloud computing. Tasks have full knowledge of the available resources and try to maximize their own utility function. Similarly, Grosu and Chronopoulos [14] formulated the load balancing problem among different users as a non-cooperative game and then studied the properties and the computation of Nash equilibria.

In this paper, the problem differs significantly from the grid computing setup of [12] or the load balancing problem of [14], [13] in cloud-computing services. In particular, we are concerned with the problem of allocating CPU resources among several applications, while applications are able to adjust their own service levels. Under the proposed scheme, both applications and the RM act independently trying to maximize their own performance measure (utility) according to a utility-based adjustment process. Naturally, this framework can be interpreted as a strategic-interaction (or game) among applications and the RM. It is shown analytically that *fair* resource allocation can be achieved in a distributed fashion through the proposed adjustment process when the CPU is overloaded. Experiments using the TrueTime Matlab toolbox show the validity of our proposed approach.

The remainder of the paper is organized as follows. The overall framework is described in Section II. The details of the resources and service levels adjustment policies are given in Section III and the convergence analysis is presented in Section IV. In Section V the approach is evaluated experimentally and Section VI presents concluding remarks.

II. FRAMEWORK

A. Resource manager & applications

The overall framework is illustrated in Figure 1. A set \mathcal{I} of n applications are competing with each other for CPU resources. Let i be a representative element of set \mathcal{I} , i.e., an application. Since we allow applications to dynamically join or leave, the number n is not constant over time.

The resources are managed by a RM making sure that the overall allocated resources does not exceed the available ones (i.e. the number of cores).

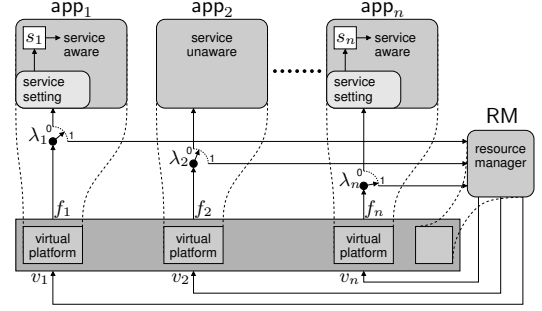


Fig. 1. Resource management framework.

The RM allocates resources to each application $i \in \mathcal{I}$ through a Constant Bandwidth Server (CBS) [2] with period P_i and budget Q_i . Hence, application i is assigned a virtual platform with bandwidth $v_i = Q_i/P_i$ corresponding to a fraction of the computing power of a single CPU. The quantity v_i can also be interpreted as speed (relative to the CPU full speed) at which the i is running. Obviously not all speeds v_i are feasible, since the sum of them cannot exceed the number m of available CPU's. Formally, we define the set of feasible speed assignments, (v_1, \dots, v_n) , as

$$\bar{\mathcal{V}} = \left\{ \mathbf{v} = (v_1, \dots, v_n) \in [0, 1]^n : \sum_{i=1}^n v_i \leq m \right\}. \quad (1)$$

Each application $i \in \mathcal{I}$ may change its *service level* $s_i \in \mathcal{S}_i \triangleq [\underline{s}_i, \infty)$, where $\underline{s}_i > 0$ is the minimum possible service level of i . Examples of service levels are: the accuracy of an iterative optimization routine, the details of an MPEG player, the sampling frequency of a controller, etc. The service level s_i is an internal state of application i . Hence, it is written/read by i only. Let also $\mathbf{s} = (s_1, \dots, s_n)$ be the service level profile of all applications evolving within $\mathcal{S} \triangleq \mathcal{S}_1 \times \dots \times \mathcal{S}_n$.

B. The matching function

The goal of the proposed resource management framework is to compute, for each application i , a matching between the service level s_i set by i and the speed v_i assigned by the RM.

Definition 2.1 (Matching function): The quality of the matching between a service level s_i and a speed v_i of application i is defined by the *matching function* $f_i : \mathcal{S}_i \times [0, 1] \rightarrow \mathbb{R}$ with the following properties:

- if $|f_i(s_i, v_i)| \leq \delta$, then the matching is *perfect*;
- if $f_i(s_i, v_i) < -\delta$, then the matching is *scarce*;
- if $f_i(s_i, v_i) > \delta$, then the matching is *abundant*.

with δ being a system parameter.

A perfect matching between s_i and v_i describes a situation in which application i has the right amount of resources v_i when it runs at service level s_i . A scarce (resp., abundant) matching describes the situation when either increasing v_i or decreasing s_i (resp., decreasing v_i or increasing s_i) is needed to move towards a perfect matching.

Notice that the service levels $\{s_i\}$ are internal states of the applications, while the virtual platforms $\{v_i\}$ belong to the RM space. Hence, neither the RM nor the applications have

a complete knowledge of the matching function f_i . In fact, the matching function is only measured during run-time. The only properties that we require from any implementation of f_i are the following.

- (P1) $s_i \neq 0 \Rightarrow f_i(s_i, 0) < -\delta$, that is, the matching must certainly be scarce if no resources are assigned;
- (P2) $s_i \geq s'_i \Rightarrow f_i(s_i, v_i) \leq f_i(s'_i, v_i)$, if application i lowers its service level, then the matching function should increase,
- (P3) $v_i \geq v'_i \Rightarrow f_i(s_i, v_i) \geq f_i(s_i, v'_i)$, if the bandwidth of application i decreases, then the matching function should also decrease.

For a real-time application, we define a *time-sensitive* matching function as follows:

$$f_i = \frac{D_i}{R_i} - 1,$$

where D_i denotes a *soft-deadline* of the application, and R_i denotes the *job response time*, that is the time elapsing from the start time to the finishing time of a job. For several classes of applications the above matching function can be rewritten with respect to s_i and v_i as follows:

$$f_i = \beta_i \frac{v_i}{s_i} - 1. \quad (2)$$

For example, for *multimedia applications*, the soft deadline D_i can be considered constant, while the response time can be defined as $R_i = C_i/v_i$, where $C_i = \alpha_i s_i$ is the execution time per job (at a service level s_i) and v_i is the speed of execution. Similarly, in *control applications*, $R_i = C_i/v_i$ where C_i denotes nominal time of execution, while the soft deadline D_i is considered inverse proportional to the sampling frequency (or service level) s_i , i.e., $D_i = \alpha_i/s_i$. Both cases lead to a matching function with the form of (2).

C. Adjustment weights

When $|f_i| > \delta$ some adjustment of the service level s_i or the virtual platform v_i is needed, in order to establish a better matching. We introduce the weight $\lambda_i \in [0, 1]$ to represent the *importance* that the RM assigns to application i when adjusting its virtual platform v_i . As we shall see in a forthcoming section, the weights $\{\lambda_i\}$ will determine the direction of adjustment of the virtual platforms $\{v_i\}$ by the RM, where applications with higher importance than others will be receiving a larger proportion of this adjustment. For the remainder of the paper, the weights $\{\lambda_i\}$ are considered given and determined by the RM.

III. ADJUSTMENT DYNAMICS

Below, we introduce a learning procedure under which the applications and the RM adapt to possible changes in their “environment” (other applications) trying to improve their own performance.

A. RM adjustment

To simplify the implementation, the RM updates the bandwidth $\tilde{v}_i = v_i/m$, normalized with respect to the number of cores. The unused bandwidth and its normalized version

are defined, respectively, by

$$v_r = m - \sum_{i=1}^n v_i, \quad \tilde{v}_r = \frac{v_r}{m} = 1 - \sum_{i=1}^n \tilde{v}_i. \quad (3)$$

At time $t = 0, 1, 2, \dots$, the RM assigns resources according to the following rule:¹

- 1) It measures performance $f_i = f_i(t)$ for each $i \in \mathcal{I}$.
- 2) It updates the normalized resource allocation vector $\tilde{\mathbf{v}} \triangleq (\tilde{v}_1, \dots, \tilde{v}_n)$ as follows:

$$\tilde{v}_i(t+1) = \Pi_{\tilde{\mathcal{V}}_i} [\tilde{v}_i(t) + \epsilon(t)g_{rm,i}(t)] \quad (4)$$

for each $i = 1, \dots, n$, where

$$g_{rm,i}(t) \triangleq -\lambda_i f_i(t) + \sum_{j=1}^n \lambda_j f_j(t) \tilde{v}_i(t),$$

and $\Pi_{\tilde{\mathcal{V}}_i}$ denotes the projection onto the feasible set $\tilde{\mathcal{V}}_i \triangleq [0, 1/m]$. The unused bandwidth is updated according to (3).

- 3) It computes the original values of bandwidths by setting $v_i(t+1) = m \tilde{v}_i(t+1)$.
- 4) It updates the time $t \leftarrow t+1$ and repeats.

We will consider a diminishing step-size sequence

$$\epsilon(t) \triangleq \frac{1}{t+1}.$$

According to recursion (4), we should expect that v_i increases when i performs poorly compared with the group of applications, i.e., when $\lambda_i f_i$ is small compared to $\sum_{j=1}^n \lambda_j f_j \tilde{v}_i$.

The above algorithm exhibits several nice properties. In particular, recursion (4) ensures that the virtual platforms $v_i(t)$ are feasible according to (1) for sufficiently large t . Furthermore, due to property (P1) and for $\lambda_i > 0$, there exists t' such that $v_i(t) > 0$ for all $t > t'$, i.e., the virtual platforms will be nonzero for all future times. Note though that this might not be the case when $\lambda_i = 0$.

In some cases, we will use vector notation for (4),

$$\tilde{\mathbf{v}}(t+1) = \tilde{\Pi}_{\{\tilde{\mathcal{V}}_i\}} [\tilde{\mathbf{v}}(t) + \epsilon(t)\mathbf{g}_{rm}(t)] \quad (5)$$

where $\mathbf{g}_{rm}(t) \triangleq -\mathbf{\Lambda}\mathbf{f}(t) + (\mathbf{1}^T \mathbf{\Lambda}\mathbf{f}(t)) \tilde{\mathbf{v}}(t)$, with $\mathbf{\Lambda} \triangleq \text{diag} \{\lambda_i\}_i$, $\mathbf{f} \triangleq [f_i]_i$ and $\tilde{\Pi}_{\{\tilde{\mathcal{V}}_i\}}[\cdot]$ denotes the combination of projections on \mathcal{V}_i 's, i.e.,

$$\tilde{\Pi}_{\{\tilde{\mathcal{V}}_i\}}[\tilde{\mathbf{v}}] \triangleq (\Pi_{\tilde{\mathcal{V}}_1}[\tilde{v}_1], \dots, \Pi_{\tilde{\mathcal{V}}_n}[\tilde{v}_n]).$$

We will use the notation $\tilde{\mathcal{V}} \triangleq \tilde{\mathcal{V}}_1 \times \dots \times \tilde{\mathcal{V}}_n$ to denote the space of $\tilde{\mathbf{v}}$.

B. Applications adjustment

The RM provides information to all applications to guide their selection of a proper service level. This information will be closely related to the performance of each application i as measured by the matching function f_i .

¹Although the performance f_i of application i is a function of both the service level s_i and the virtual platform v_i , the RM simply receives an instance of this function. Thus, we abuse notation by simply writing $f_i = f_i(t)$.

We will consider an adjustment process for the service levels of each application which has the generic form:

$$s_i(t+1) = \Pi_{\mathcal{S}_i} [s_i(t) + \epsilon(t)g_{\text{app},i}(t)], \quad (6)$$

where the term $g_{\text{app},i}(t)$ captures an “observation” sent by the RM to the application and depends on the matching function f_i . In particular, we would like this observation to be zero when the matching function is zero.

In several cases, we will use the more compact form:

$$\mathbf{s}(t+1) = \tilde{\Pi}_{\{\mathcal{S}_i\}} [\mathbf{s}(t) + \epsilon(t)\mathbf{g}_{\text{app}}(t)]. \quad (7)$$

where $\mathbf{s} \triangleq [s_i]_i$ and $\mathbf{g}_{\text{app}} \triangleq [g_{\text{app},i}]_i$.

Below, we identify two candidates for the observation term $g_{\text{app},i}$.

1) *Scheme (a)*: The first scheme is rather generic and independent of the specific form of the matching function f_i . It simply defines $g_{\text{app},i}(t) \triangleq f_i(t)$. Naturally, in this case, the specifications we set above are satisfied.

2) *Scheme (b)*: The second scheme takes into account the specific form of the matching function (2). Assuming that application i could read $v_i(t)$, a natural way for the application to adjust its service level is to simply set $s_i(t+1) = \beta_i v_i(t)$, since, according to (2), if s_i matches $\beta_i v_i$, the matching function will become zero. However, setting the next service level $s_i(t+1)$ according to this rule relies on a careful estimation of β_i , which may be unavailable. From the (possibly non-zero) measurement $f_i(t)$ at time t we can estimate β_i as

$$\beta_i = (1 + f_i(t)) \frac{s_i(t)}{v_i(t-1)},$$

in which case, the service level update rule becomes

$$s_i(t+1) = (1 + f_i(t)) \frac{v_i(t)}{v_i(t-1)} s_i(t). \quad (8)$$

The above recursion may exhibit large incremental differences, $s_i(t+1) - s_i(t)$, which is not desirable. Hence, we introduce a smoother update rule for the service level s_i that exhibits the same stationary points of (8), by setting:

$$g_{\text{app},i}(t) \triangleq (1 + f_i(t)) \frac{v_i(t)}{v_i(t-1)} s_i(t) - s_i(t). \quad (9)$$

In words, we should expect that s_i decreases when $f_i < 0$ and $v_i(t)/v_i(t-1) < 1$, i.e., when the application is doing poorly and the assigned resources have been decreased. The term $v_i(t)/v_i(t-1)$ provides a look-ahead information to the application about the expectation over future resources.

C. Resource allocation game

Briefly, we would like to note that the above framework naturally introduces a strategic-form game (cf., [15]) between the applications and the RM. Note that a strategic-form game is defined as a collection of: (i) a set of players \mathcal{P} , which here is defined as the set of applications \mathcal{I} and the RM, (ii) a set of available actions, \mathcal{A}_p , for each player $p \in \mathcal{P}$, which is defined as the set of available service levels for an application i or the set of available virtual platforms for the RM; and (iii) a set of utilities or performance measures, $u_p : \mathcal{A}_p \rightarrow \mathbb{R}$, for each player p . The selection of these utility

functions is open-ended. A candidate selection, motivated by the dynamics presented above could be as follows:

– for each $i \in \mathcal{I}$, $u_{\text{app},i} : \mathcal{S}_i \times [0, 1] \rightarrow \mathbb{R}$, such that

$$\nabla_{s_i} u_{\text{app},i}(s_i, v_i) = f_i(s_i, v_i);$$

– for the RM, $u_{\text{rm}} : \mathcal{S} \times \bar{\mathcal{V}} \rightarrow \mathbb{R}$, such that, for each i ,

$$\nabla_{v_i} u_{\text{rm}}(\mathbf{s}, \mathbf{v}) = -\lambda_i f_i(s_i, v_i) + \sum_{j=1}^n \lambda_j f_j(s_j, v_j) \tilde{v}_i.$$

Under such strategic-form game formulation, each application would prefer to select s_i that makes the matching function f_i equal to zero, while the RM would prefer to select a virtual platform allocation that would be “fair” to all applications. Furthermore, under this framework the adjustment dynamics presented before introduce a natural way for searching for a Nash equilibrium allocation (cf. [15]).

IV. CONVERGENCE ANALYSIS

In this section, we analyze the asymptotic behavior of the adjustment dynamics of the RM (5) and the applications (7). For the remainder of the paper, we will consider scheme (b) for the applications adjustment presented in Section III-B.

For the sake of analysis, we will abuse notation by taking the observation signals $g_{\text{rm},i}$ and $g_{\text{app},i}$ as functions of the service levels and virtual platforms. In particular, for the RM adjustment, the observation signal $g_{\text{rm},i}(t)$ is replaced by $g_{\text{rm},i} : \mathcal{S} \times \bar{\mathcal{V}}$ such that:

$$g_{\text{rm},i}(\mathbf{s}, \tilde{\mathbf{v}}) \triangleq -\lambda_i f_i(s_i, v_i) + \sum_{j=1}^n \lambda_j f_j(s_j, v_j) \tilde{v}_i,$$

where $v_i = m \tilde{v}_i$. Likewise, the observation signal $g_{\text{app},i}(t)$ in the application i 's adjustment is replaced by the function $g_{\text{app},i} : \mathcal{S}_i \times \tilde{\mathcal{V}}_i$ such that:

$$g_{\text{app},i}(s_i, \tilde{v}_i, y_i) = (1 + f_i(s_i, v_i)) \frac{\tilde{v}_i(t)}{y_i(t)} s_i(t) - s_i(t),$$

where

$$y_i(t+1) = y_i(t) + \epsilon(t) (\tilde{v}_i(t) - y_i(t)). \quad (10)$$

The reason for introducing the new state variable $\mathbf{y} \triangleq (y_1, \dots, y_n)$ is to deal with the different time indices in (9). The state variable \mathbf{y} represents a low-pass filter of $\tilde{\mathbf{v}}$, and for sufficiently large time t remains positive due to (P1).

The asymptotic behavior of the overall adjustment described by (5), (7) and (10), can be characterized as follows:

Proposition 4.1: The overall recursion (5), (7) and (10) is such that the sequence $\{(\mathbf{s}(t), \tilde{\mathbf{v}}(t), \mathbf{y}(t))\}$ converges² to some limit set of the ODE:

$$(\dot{\mathbf{s}}, \dot{\tilde{\mathbf{v}}}, \dot{\mathbf{y}}) = \mathbf{g}(\mathbf{s}, \tilde{\mathbf{v}}, \mathbf{y}) + \mathbf{z}(t), \quad (11)$$

where $\mathbf{g} \triangleq (\mathbf{g}_{\text{app}}, \mathbf{g}_{\text{rm}}, \tilde{\mathbf{v}} - \mathbf{y})$ and $\mathbf{z} \triangleq (\mathbf{z}_{\text{app}}, \mathbf{z}_{\text{rm}}, 0)$ is the minimum force required to drive $\tilde{v}_i(t)$ to \mathcal{V}_i and $s_i(t)$ back to \mathcal{S}_i . Finally, if $E \subset \mathcal{S} \times \bar{\mathcal{V}} \times [0, 1]^n$ is a locally asymptotically stable set in the sense of Lyapunov³ for (11) and $(\mathbf{s}(t), \tilde{\mathbf{v}}(t), \mathbf{y}(t))$ is in some compact set in the domain of attraction of E , then $(\mathbf{s}(t), \tilde{\mathbf{v}}(t), \mathbf{y}(t)) \rightarrow E$.

²By $x(t) \rightarrow A$ for a set A , we mean $\lim_{t \rightarrow \infty} \text{dist}(x(t), A) = 0$.

³See [16, Definition 3.1].

Proof: The proof is based on Theorem 2.1 in [17]. ■

The above proposition relates the asymptotic behavior of the overall discrete-time recursion with the limit sets of the ODE (11). Since the *stationary points*⁴ of the vector field \mathbf{g} are invariant sets of the ODE (11), then they are also candidate attractors for the recursion. In the following sections, we analyze the convergence properties of the recursion with respect to the stationary points of the ODE (11).

A. Stationary points

Lemma 4.1 (Stationary Points): Any stationary point of the ODE (11) satisfies all the following conditions:

- (C1) $\sum_i \lambda_i f_i(s_i^*, v_i^*) \tilde{v}_j^* = \lambda_j f_j(s_j^*, v_j^*)$, or $\{\tilde{v}_j^* = 1/m \wedge f_j(s_j^*, v_j^*) \leq 0\}$;
- (C2) $f_j(s_j^*, v_j^*) = 0$, or $\{s_j^* = \underline{s}_j \wedge f_j(s_j^*, v_j^*) \leq 0\}$,
- (C3) $y_j^* = \tilde{v}_j^*$,

for all $j \in \mathcal{I}$. Furthermore, the set of stationary points is non-empty.

Proof: Condition (C1) is an immediate consequence of setting $g_{rm,j}(s^*, \tilde{v}^*) + z_{rm,j} = 0$, $j \in \mathcal{I}$. Likewise, conditions (C2) and (C3) follow directly from setting $g_{app,j}(s^*, \tilde{v}^*, y_i) + z_{rm,j} = 0$ and $y_j = \tilde{v}_j$, $j \in \mathcal{I}$.

Regarding existence of stationary points, and without loss of generality, we restrict attention to the allocations $(\mathbf{s}, \tilde{\mathbf{v}})$ for which $f_i(s_i, v_i) \leq 0$ for all $i \in \mathcal{I}$ (since, if there exists application i for which $f_i(s_i, v_i) > 0$, then i may always increase s_i to match v_i without affecting the matching functions of the other applications). Under this restriction, we consider two cases: (a) there exists $\mathbf{s}^* \in \mathcal{S}$ and $\tilde{\mathbf{v}}^* \in \tilde{\mathcal{V}}$ such that $f_j(s_j^*, v_j^*) = 0$ for all $j \in \mathcal{I}$; and (b) there exists at least one $j \in \mathcal{I}$ such that $f_j(s_j, v_j) < 0$ for all $s_j \in \mathcal{S}_j$ and $\tilde{v}_j \in \tilde{\mathcal{V}}_j$. In case (a), $(\mathbf{s}^*, \tilde{\mathbf{v}}^*, \tilde{\mathbf{v}}^*)$ is a stationary point of the ODE (11). In case (b), $\sum_i \lambda_i f_i(s_i, v_i) < 0$ for the allocations under consideration. Then, condition (C1) gives:

$$\tilde{\mathbf{v}}^* = \mathbf{h}(\mathbf{s}^*, \tilde{\mathbf{v}}^*) \triangleq \frac{\mathbf{\Lambda} \mathbf{f}(\mathbf{s}^*, \mathbf{v}^*)}{\mathbf{1}^T \mathbf{\Lambda} \mathbf{f}(\mathbf{s}^*, \mathbf{v}^*)}, \quad (12)$$

where $\mathbf{v}^* = m \tilde{\mathbf{v}}^*$. Since the function $\mathbf{h}(\mathbf{s}^*, \cdot)$ is continuous on a convex and compact set $\tilde{\mathcal{V}}$, by Brower's fixed point theorem (cf., [18, Corollary 6.6]) $\mathbf{h}(\mathbf{s}^*, \cdot)$ has a fixed point. If the fixed point suggests $\tilde{v}_j^* > 1/m$ for all $j \in J \subseteq \mathcal{I}$, then set $\tilde{v}^* \equiv 1/m$ for all $j \in J$ and resolve (12) to compute a stationary point for the rest of applications $\mathcal{I} \setminus J$. This process will define a stationary point. ■

In words, the above lemma states that at a stationary point, application i is either performing sufficiently good (i.e., $f_j(s_j^*, v_j^*) = 0$), or it performs poorly but i) its service level s_i cannot be decreased any further (i.e., $f_j(s_j^*, v_j^*) \leq 0$, $s_j^* = \underline{s}_j$), and/or ii) its virtual platform v_i cannot be increased any further (i.e., $f_j(s_j^*, v_j^*) \leq 0$, $\tilde{v}_j^* = 1/m$).

Note that any pair $(\mathbf{s}, \tilde{\mathbf{v}})$ for which $f_i(s_i, v_i) = 0$ is a stationary point of the ODE (11). This multiplicity of stationary points complicates the convergence analysis, however, in some cases, uniqueness of the stationary points can be shown. Next, we analyze one such special case.

1) Fixed Service Levels: The following result characterizes the set of stationary points when the service levels are *fixed*, i.e., when each application has only one service level. First, define the following constants:

$$\Theta_s \triangleq \inf_{\tilde{\mathbf{v}} \in \tilde{\mathcal{V}}} \left| \sum_i \lambda_i f_i(s_i, v_i) \right| \geq 0,$$

$$K_s \triangleq \sup_{i \in \mathcal{I}, \tilde{\mathbf{v}} \in \tilde{\mathcal{V}}} |f_i(s_i, v_i)| < \infty,$$

$$\gamma_s \triangleq \sup_{i \in \mathcal{I}} \left\{ \frac{\beta_i m}{s_i} \right\} > 0.$$

Proposition 4.2 (Uniqueness of Stationary Points): For some given $\mathbf{s} \in \mathcal{S}$, let f_i be defined by (2). If $\Theta_s > 0$ and

$$\rho_s \triangleq \frac{K_s \gamma_s (\sum_i \lambda_i)^2}{\Theta_s^2} < 1,$$

then, the ODE (11) exhibits a unique stationary point.

Proof: We first consider the unconstrained case where $\tilde{\mathcal{V}}_i = [0, 1]$ for each $i \in \mathcal{I}$, i.e., there is only one core ($m = 1$). (We will revisit this assumption later.) Given also that $\Theta_s > 0$, a stationary point $\tilde{\mathbf{v}}^*$ satisfies $\tilde{\mathbf{v}}^* = \mathbf{h}(\mathbf{s}, \tilde{\mathbf{v}}^*)$ for some constant vector \mathbf{s} .

For any $\tilde{\mathbf{v}}' \neq \tilde{\mathbf{v}}$, we have,

$$\begin{aligned} & h_j(\mathbf{s}, \tilde{\mathbf{v}}') - h_j(\mathbf{s}, \tilde{\mathbf{v}}) \\ &= \frac{\lambda_j f_j(s_j, v_j')}{\sum_i \lambda_i f_i(s_i, v_i')} - \frac{\lambda_j f_j(s_j, v_j)}{\sum_i \lambda_i f_i(s_i, v_i)} \end{aligned}$$

Note that, $|\sum_i \lambda_i f_i(s_i, v_i')| \geq \Theta_s^2 > 0$. Thus,

$$|h_j(\mathbf{s}, \tilde{\mathbf{v}}') - h_j(\mathbf{s}, \tilde{\mathbf{v}})| \leq \frac{\lambda_j}{\Theta_s^2} \sum_i \lambda_i |f_j(s_j, v_j') f_i(s_i, v_i) - f_j(s_j, v_j) f_i(s_i, v_i')|.$$

Also, we have:

$$\begin{aligned} & |f_j(s_j, v_j') f_i(s_i, v_i) - f_j(s_j, v_j) f_i(s_i, v_i')| \\ & \leq K_s (|f_j(s_j, v_j') - f_j(s_j, v_j)| + |f_i(s_i, v_i) - f_i(s_i, v_i')|) \\ & \leq K_s \left(\frac{\beta_j m}{s_j} |\tilde{v}_j' - \tilde{v}_j| + \frac{\beta_i m}{s_i} |\tilde{v}_i - \tilde{v}_i'| \right) \\ & \leq K_s \gamma_s \|\tilde{\mathbf{v}}' - \tilde{\mathbf{v}}\|_1 \end{aligned}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm in \mathbb{R}^n . We conclude that

$$|h_j(\mathbf{s}, \tilde{\mathbf{v}}') - h_j(\mathbf{s}, \tilde{\mathbf{v}})| \leq \frac{\lambda_j}{\Theta_s^2} \sum_i \lambda_i K_s \gamma_s \|\tilde{\mathbf{v}}' - \tilde{\mathbf{v}}\|_1,$$

which also implies that

$$\|\mathbf{h}(\mathbf{s}, \tilde{\mathbf{v}}') - \mathbf{h}(\mathbf{s}, \tilde{\mathbf{v}})\|_1 \leq \frac{K_s \gamma_s (\sum_i \lambda_i)^2}{\Theta_s^2} \cdot \|\tilde{\mathbf{v}}' - \tilde{\mathbf{v}}\|_1.$$

Hence, \mathbf{h} defines a contraction if $\rho_s < 1$. Therefore, by Banach Fixed Point Theorem (cf., [19, Theorem 5.1-2]) and the fact that ℓ_1 is a complete metric space in \mathbb{R}^n , \mathbf{h} has a unique fixed point.

If, instead, $\tilde{\mathcal{V}}_i = [0, 1/m]$, for all $i \in \mathcal{I}$, we proceed as follows: We set $\tilde{v}_j^* = 1/m$ for all $j \in J \subseteq \mathcal{I}$ for which the unconstrained solution suggests $\tilde{v}_j^* > 1/m$. Then, we repeat the same steps as in the unconstrained case for all $i \in \mathcal{I} \setminus J$. ■

The following corollary discusses one special case where uniqueness of stationary points can be shown.

⁴The stationary points of an ODE $\dot{x} = \mathbf{g}(x)$ are defined as the points in the domain D for which $\mathbf{g}(x) = 0$.

Corollary 4.1 (Large service levels): Consider the matching function defined by (2). For some given $\mathbf{s} \in \mathcal{S}$, let us assume that $\beta_i/s_i \ll 1$ for all i . Then, the ODE (5) has a unique stationary point $\tilde{\mathbf{v}}^*$. Furthermore, as $\beta_i/s_i \rightarrow 0$ for all i , then

$$\tilde{v}_i^* \rightarrow \min \left\{ \frac{1}{m}, \frac{\lambda_i}{\sum_j \lambda_j} \right\}, \quad \forall i \in \mathcal{I}. \quad (13)$$

Proof: For the matching function of (2), we have:

$$\Theta_s = \inf_{\tilde{\mathbf{v}} \in \tilde{\mathcal{V}}} \left| \sum_i \lambda_i \frac{\beta_i m}{s_i} \tilde{v}_i - \sum_i \lambda_i \right|.$$

If $\beta_i/s_i \ll 1$ for all i , then there exists $i^* \in \mathcal{I}$ such that

$$\Theta_s = \sum_i \lambda_i - \lambda_{i^*} \frac{\beta_{i^*}}{s_{i^*}} \gg 0.$$

Also, $K_s = \sup_{i \in \mathcal{I}, \tilde{\mathbf{v}} \in \tilde{\mathcal{V}}} |f_i(s_i, v_i)| = 1$. Thus, we have:

$$\rho_s = \frac{\gamma_s \left(\sum_i \lambda_i \right)^2}{\left(\sum_i \lambda_i - \lambda_{i^*} \frac{\beta_{i^*}}{s_{i^*}} \right)^2}.$$

For $\beta_i/s_i \ll 1$ for all i , we have $\gamma_s \ll 1$, which further implies that $\rho_s < 1$. Therefore, from Proposition 4.2, there exists a unique stationary point.

Furthermore, as $\beta_i/s_i \rightarrow 0$, the stationary point can be computed from condition (C1) and it can be verified in a straightforward manner that satisfies (13). ■

The above proposition also provides an answer to how the stationary point changes with respect to the weight parameters $\{\lambda_i\}$ for large service levels. In that case, we conclude that the larger the weight λ_i assigned by the RM to application i , the larger the asymptotic bandwidth v_i^* .

2) *Non-fixed Service Levels:* When the service levels are also adjusted according to (7), the characterization of the set of stationary points is not straightforward. Motivated by Corollary 4.1 the following corollary identifies one case where uniqueness of stationary points holds.

Corollary 4.2 (Overloaded CPU): Consider the matching function defined by (2). If the solution of the ODE (11), $\{\tilde{\mathbf{v}}(t)\}$, satisfies $\sup_{t \geq 0} \{\beta_i m \tilde{v}_i(t)/s_i\} \ll 1$ for all i , then the ODE (11) exhibits a unique stationary point $(\mathbf{s}^*, \tilde{\mathbf{v}}^*)$ such that $s_i^* = s_i$. Furthermore, as $\sup_{t \geq 0} \{\beta_i m \tilde{v}_i(t)/s_i\} \rightarrow 0$ for all i , then \tilde{v}_i^* satisfies property (13).

Proof: (sketch) If the solution of the ODE (11) satisfies $\sup_{t \geq 0} \{\beta_i v_i(t)/s_i\} \ll 1$ for all i , i.e., the virtual platforms are restricted within small values, then we may repeat the analysis of Corollary 4.1, for the restricted set of feasible virtual platforms to show that $\rho_s < 1$ for all $s \in \mathcal{S}$. ■

The hypotheses of Corollary 4.2 are not restrictive. In fact, when the CPU is overloaded, i.e., large number of applications is currently running, the amount of resources that the RM assigns to each application will be small at all times leading to $\sup_{t \geq 0} \{\beta_i m \tilde{v}_i(t)/s_i\} \ll 1$. In such cases, the RM is treating applications fairly, as property (13) indicates.

As we have already pointed out, in the more general case where the hypotheses of Corollary 4.2 do not hold, there is a multiplicity of stationary points including (if exist) any pair $(\mathbf{s}^*, \tilde{\mathbf{v}}^*)$ for which $f_i(s_i^*, v_i^*) = 0$ for all i .

B. Local Asymptotic Stability & Convergence

The following proposition characterizes locally the stability properties of the stationary points under the hypotheses of Corollary 4.2.

Proposition 4.3 (LAS): Under the hypotheses of Corollary 4.2, the unique stationary point of the dynamics (11) is a locally asymptotically stable point in the sense of Lyapunov.

Proof: Let $\tilde{\mathbf{v}}^*$ corresponds to the unique stationary point of (11). Define the non-negative function

$$W(\tilde{\mathbf{v}}) = \frac{1}{2} (\tilde{\mathbf{v}} - \tilde{\mathbf{v}}^*)^T (\tilde{\mathbf{v}} - \tilde{\mathbf{v}}^*) \geq 0$$

and consider the unconstrained case at which $\tilde{\mathcal{V}}_i = [0, 1]$ for all $i \in \mathcal{I}$. (We will revisit this assumption later on.) Then,

$$\begin{aligned} \dot{W}(\tilde{\mathbf{v}}) &= (\tilde{\mathbf{v}} - \tilde{\mathbf{v}}^*)^T \mathbf{g}_{\text{rm}}(\mathbf{s}, \tilde{\mathbf{v}}) \\ &= (\tilde{\mathbf{v}} - \tilde{\mathbf{v}}^*)^T [-\mathbf{A}\mathbf{f}(\mathbf{s}, \mathbf{v}) + \mathbf{1}^T \mathbf{A}\mathbf{f}(\mathbf{s}, \mathbf{v}) \tilde{\mathbf{v}}]. \end{aligned}$$

Let us consider the following perturbed allocation $\tilde{\mathbf{v}} = (1 - \varepsilon)\tilde{\mathbf{v}}^* + \varepsilon \mathbf{w}$ for some $\mathbf{w} \in \tilde{\mathcal{V}}$ and $\varepsilon > 0$. Then, we have:

$$\dot{W}(\tilde{\mathbf{v}}) = \varepsilon (\mathbf{w} - \tilde{\mathbf{v}}^*)^T [-\mathbf{A}\mathbf{f}(\mathbf{s}, (1 - \varepsilon)m\tilde{\mathbf{v}}^* + \varepsilon m\mathbf{w}) + \mathbf{1}^T \mathbf{A}\mathbf{f}(\mathbf{s}, (1 - \varepsilon)m\tilde{\mathbf{v}}^* + \varepsilon m\mathbf{w})((1 - \varepsilon)\tilde{\mathbf{v}}^* + \varepsilon \mathbf{w})].$$

It is also straightforward to verify that:

$$\mathbf{f}(\mathbf{s}, (1 - \varepsilon)m\tilde{\mathbf{v}}^* + \varepsilon m\mathbf{w}) = (1 - \varepsilon)\mathbf{f}(\mathbf{s}, m\tilde{\mathbf{v}}^*) + \varepsilon \mathbf{f}(\mathbf{s}, m\mathbf{w}).$$

Thus,

$$\begin{aligned} \dot{W}(\tilde{\mathbf{v}}) &\approx \varepsilon^2 \|\mathbf{w} - \tilde{\mathbf{v}}^*\|_2^2 \mathbf{1}^T \mathbf{A}\mathbf{f}(\mathbf{s}, m\tilde{\mathbf{v}}^*) + \\ &\quad \varepsilon^2 (\mathbf{w} - \tilde{\mathbf{v}}^*)^T [-\mathbf{A}\mathbf{f}(\mathbf{s}, m\mathbf{w}) + \mathbf{1}^T \mathbf{A}\mathbf{f}(\mathbf{s}, m\mathbf{w}) \tilde{\mathbf{v}}^*] \end{aligned}$$

plus higher order terms of ε .

If the hypotheses of Corollary 4.2 hold, i.e., $\sup_{t \geq 0} \{\beta_i m \tilde{v}_i(t)/s_i\} \ll 1$ for all i , then the first term of the RHS of the above expression dominates the second term. This is due to the fact that as $\sup_{t \geq 0} \{\beta_i m \tilde{v}_i(t)/s_i\}$ approaches zero for all i , the second term approaches zero while the first term is bounded away from zero and is strictly negative. Therefore, from [16, Theorem 3.1], the stationary point $\tilde{\mathbf{v}}^*$ is locally asymptotically stable.

Finally, in case $\tilde{\mathcal{V}}_i = [0, 1/m]$, the unique stationary point may assign $\tilde{v}_i^* = 1/m$ for some applications i . In this case, it is straightforward to check that the vector field \mathbf{g}_{rm} points outwards, which implies that the conclusions of the unconstrained case continue to hold. ■

From Proposition 4.1, we conclude that the stationary points of the ODE (11), which satisfy the hypotheses of Proposition 4.3, are local attractors of the overall recursion.

V. EXPERIMENTAL EVALUATION

To investigate the assignment of the bandwidth and the values of the application service levels, the resource management scheme was implemented both in Matlab and in TrueTime [20].

A. Experiment with synthetic applications

In the first Matlab experiment, the applications are not executed, but they are simply abstracted by their characteristic parameters. We have three applications running over two cores. Applications are all the same, except for the values of the weights: $\lambda_1 = 0.9$, $\lambda_2 = 0.5$ and $\lambda_3 = 0.1$. As explained

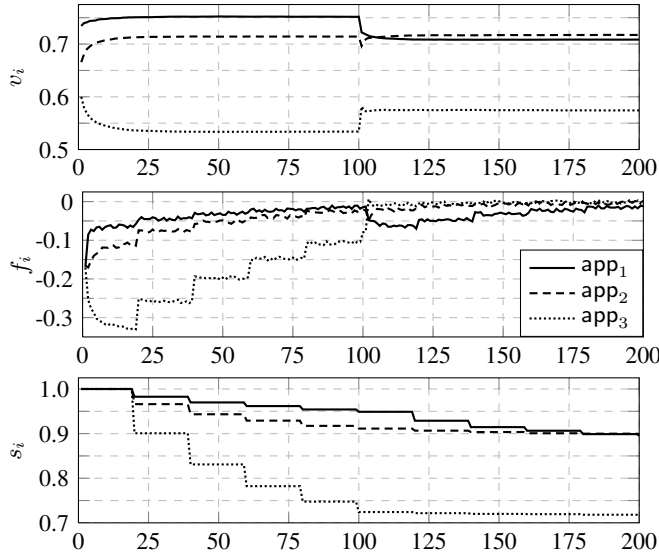


Fig. 2. Simulation results of three applications over two cores.

in Section II-C, λ_i determines the amount of effort that is taken by the RM to achieve a perfect matching for application i (i.e., f_i close to zero).

In accordance with the time-sensitive application model of Section II-B, each application is set to have $D_i = 2500$ and $\alpha_i = 2000$. In Figure 2, we demonstrate how the bandwidth v_i , the matching function f_i , and the service level s_i of all three applications evolve under the adjustment dynamics of (5), (7) and (10). Some noise is added to the matching functions f_i , to account for the inaccuracy of the real measures and to show the robustness of the method. Also, the service level adaptation is performed once every twenty steps of the RM execution, to resemble some real behavior, where applications are adjusting at a slower rate with respect to the resource allocation. At time 100, the weights of the applications are changed to $\lambda_1 = 0.1$, $\lambda_2 = 0.5$ and $\lambda_3 = 0.9$ and the RM is reinitialized.

At time 0, in response to an equally scarce matching between the bandwidth and the service levels, app_1 is assigned more resource, while app_3 has to significantly lower its service level s_3 . This observation complies with the prediction of Corollary 4.2 and Proposition 4.3, however we should not expect to observe the exact allocation (13), since the hypotheses of Corollary 4.2 partially hold at the beginning of the simulation (when $f_i \ll 0$). As the weights are changed by the RM, app_3 receives more resource, but not as predicted by (13) since the matching function is already quite close to zero.

B. Experiments with real applications

TrueTime [20] is a Matlab/Simulink-based tool that allows simulation of tasks executing within real-time kernels and communication over networks, embedded within Simulink. Among other things, it supports simulation of CBS-based [2] task execution. The policy allows to adjust the CPU time allocated to the running applications, in the same exact way

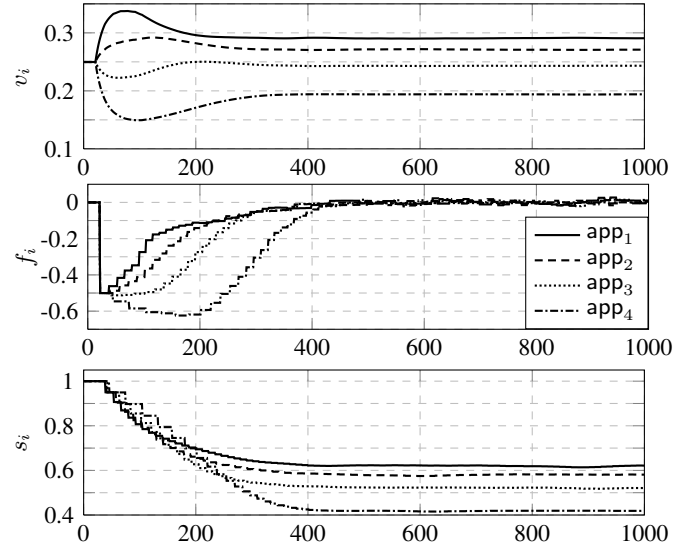


Fig. 3. TrueTime simulation results of a single core machine with four applications.

as in a real-time computing system. Moreover, TrueTime offers the ability to simulate memory management and protection, therefore being a perfect match to simulate our resource management framework.

A TrueTime kernel simulates a single CPU that hosts the execution of the RM and the CBS servers that contain the applications. A shared memory segment is initialized and both the RM and the applications have access to the memory area reporting their execution data. The RM reads the matching function, f_i , of each application i and computes the new reservations v_i . Then, it updates the parameters of the CBS server and writes in the shared memory values (to be read by the applications). The execution time of the RM is a parameter of the simulation. Both the applications and the RM are coded in a way that is very close to a real implementation and the resulting simulation data are generally very close to the data that would have been obtained on a real execution platform.

The first experiment with TrueTime considers four applications and the RM, which employ the adjustment process of (5), (7) and (10). Figure 3 shows the allocated bandwidths v_i , the matching functions f_i and the service levels s_i . For these applications, we take $D_i/\alpha_i = 2$ as explained in the derivation of (2) for multimedia applications. The weights are $\lambda_1 = 0.8$, $\lambda_2 = 0.6$, $\lambda_3 = 0.4$ and $\lambda_4 = 0.2$. Some randomness is also introduced in the execution times to show the effect of disturbances generated by lock acquisition, resource contention, memory management, etc. At the beginning of the simulation and when the matchings are quite scarce (i.e., the hypotheses of Corollary 4.2 are satisfied), the RM distributes the CPU as predicted by Corollary 4.2 and Proposition 4.3. However, as the service levels also adjust and the matching functions approach zero, we observe a deviation from the exact resource allocation predicted by (13), which is anticipated since the hypotheses

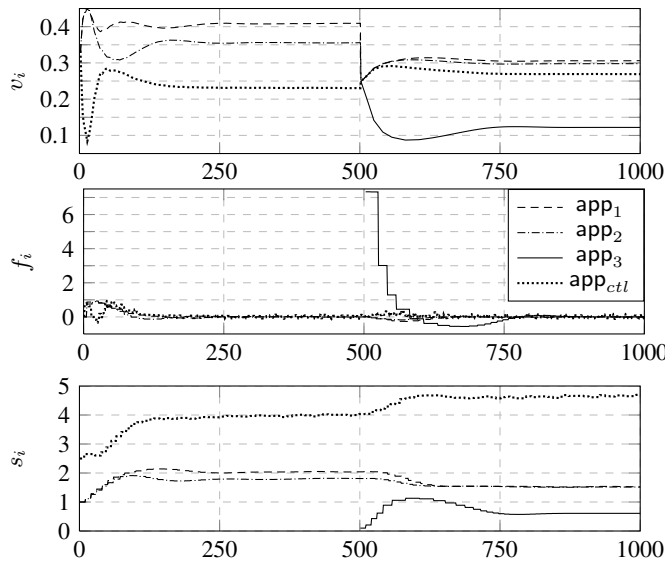


Fig. 4. TrueTime simulation results of a single core hardware with two applications and a control task. Another application is arriving at time 500.

of Corollary 4.2 no longer apply.

In the second experiment, we show how four heterogeneous applications are handled by the RM. Three of them are active from the very beginning while the fourth one enters the system at time 500. Among the three applications that are active from the beginning, there is an LQG controller controlling an inverted pendulum which is simulated in Simulink. The controller, modeled as a time-sensitive application (2), has a deadline set to $0.85T_s$ where T_s is its sampling period. Its service level s_i is simply set equal to $1/T_s$, because of the natural observation that faster sampling can provide better performance. The applications' weights are set as $\lambda_1 = 0.64$, $\lambda_2 = 0.73$, $\lambda_3 = 0.41$ and $\lambda_{ctl} = 0.41$. Every five controller jobs, the service level of the controller, i.e., the sampling frequency, is adjusted. A new sampling period is chosen and the controller is redesigned taking into account the measured sampling period and the measured input/output latency. The latency, i.e., the average amount of time between the sensor measurement and the actuation, depends on the amount of bandwidth assigned to the controller by the resource manager. In a real system the on-line redesign would be replaced by look-up in a table consisting of pre-calculated controller parameters for different sampling periods and latencies.

Figure 4 shows how the CPU bandwidth, the performance functions and the service levels evolve. Notice that when a new application is introduced the resource manager is reset.

VI. CONCLUSIONS

We proposed a distributed management framework for allocating CPU resources to time-sensitive applications. According to the proposed scheme, a resource manager is responsible for assigning resources (virtual platforms) to applications, aiming at a "fair" allocation, while each application is responsible for adjusting its own service level, aiming at a better "matching" between its service level and

the provided bandwidth. By distributing the adjustment of the virtual platforms and the service levels, the proposed scheme exhibits linear time complexity in the number of demanding applications. We analyzed the convergence behavior of the adjustment dynamics. Especially for the case of an overloaded CPU, we showed that the adjustment process assigns resources fairly among applications. We also validated the results experimentally using the TrueTime Matlab toolbox.

REFERENCES

- [1] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves: Operating system support for multimedia applications," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Boston, MA, U.S.A., May 1994, pp. 90–99.
- [2] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain, Dec. 1998, pp. 4–13.
- [3] D. C. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A feedback-driven proportion allocator for real-rate scheduling," in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, Feb. 1999.
- [4] J. Eker, P. Hagander, and K.-E. Årzén, "A feedback scheduler for real-time controller tasks," *Control Engineering Practice*, vol. 8, no. 12, pp. 1369–1378, Jan. 2000.
- [5] T. Cucinotta, L. Palopoli, L. Abeni, D. Faggioli, and G. Lipari, "On the integration of application level and resource level qos control for real-time applications," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 479–491, Nov. 2010.
- [6] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *Proceedings of the IEEE Real Time System Symposium*, 1997.
- [7] C. Lee, J. P. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource QoS problem," in *Proceedings of the 20th IEEE Real-Time Systems Symposium*, Phoenix, AZ, Dec. 1999, pp. 315–326.
- [8] M. Sojka, P. Piša, D. Faggioli, T. Cucinotta, F. Checconi, Z. Hanzálek, and G. Lipari, "Modular software architecture for flexible reservation mechanisms on heterogeneous resources," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 366–382, 2011.
- [9] E. Bini, G. C. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K.-E. Årzén, R. Vanessa, and C. Scordino, "Resource management on multicore systems: The ACTORS approach," *IEEE Micro*, vol. 31, no. 3, pp. 72–81, 2011.
- [10] B. Johansson, C. Adam, M. Johansson, and R. Stadler, "Distributed resource allocation strategies for achieving quality of service in server clusters," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 1990–1995.
- [11] K.-E. Årzén, V. Romero Segovia, S. Schorr, and G. Fohler, "Adaptive resource management made real," in *Proc. 3rd Workshop on Adaptive and Reconfigurable Embedded Systems*, Chicago, IL, USA, Apr. 2011.
- [12] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," *IEEE Transactions on Computers*, vol. 57, no. 10, pp. 1413–1422, Oct. 2008.
- [13] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, Nov. 2010.
- [14] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1022–1034, 2005.
- [15] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA: MIT Press, 1994.
- [16] H. Khalil, *Nonlinear Systems*. Prentice-Hall, 1992.
- [17] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed. Springer-Verlag New York, Inc., 2003.
- [18] K. Border, *Fixed Point Theorems with Applications to Economics and Game Theory*. Cambridge University Press, 1985.
- [19] E. Kreyszig, *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.
- [20] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Systems Magazine*, vol. 23, no. 3, p. 1630, Jun. 2003.