Stability Conditions of On-line Resource Managers for Systems with Execution Time Variations

Sergiu Rafiliu, Petru Eles, Zebo Peng
Department of Computer and Information Science,
Linköping University, Sweden
{serra,petel,zpe} @ida.liu.se

***Abstract*—Today's embedded systems are exposed to variations in load demand due to complex software applications, hardware platforms, and impact of the run-time environments. When these variations are large, and efficiency is required, on-line resource managers may be deployed on the system to help it control its resource usage. An often neglected problem is whether these resource managers are stable, meaning that the resource usage is controlled under all possible scenarios. In this paper we develop mathematical models for the real-time embedded system and we derive conditions which, if satisfied, lead to stable systems. For the developed system models, we also determine bounds on the worst case response times of tasks.**

I. Introduction and Related Work

Today's embedded systems, together with the real-time applications running on them, have achieved a high level of complexity. Moreover, such systems very often are exposed to varying resource demand (load demand) due to e.g. variable number of tasks in the system or variable execution times of tasks. When these variations are large and system efficiency is required, on-line resource managers may be deployed, to control the system's resource usage. Such managers take the shape of algorithms which run at key moments in time and adjust system parameters (task rates, modes, offsets, priorities, etc.) subject to the measured variations. Among the goals of such a resource manager is to maximize the resource usage while minimizing the amount of time the system spends in overload situations.

One, often overlooked, question is whether the deployed resource managers are safe, meaning that the resource demand is bounded under all possible runtime scenarios. This notion of safety can be linked with the notion of stability of control systems. In control applications, a controller controls the state of a plant towards a desired stationary point. The combined system is stable if the plant's state remains within a bounded distance from the stationary point, under all possible scenarios. By modeling the real-time system as the plant and the resource manager as the controller, one may be able to reason about the stability of the combined system.

When considering resource managers to control utilization, the existing literature can be classified into two categories: ad-hoc approaches and control theory based approaches. There is a vast literature on ad-hoc approaches, targeting different types of real-time systems. Lee et al. proposed the QRAM algorithm in [1]. The model consists of a number of resources that tasks use and a number of quality dimensions. When the algorithm runs, it optimizes the overall quality subject to keeping the resource constraints.

Buttazzo et al. [2] introduced the elastic model where each task's rate can change within a certain interval. Rates change when a task is added or removed from the system. Further work deals with unknown and variable execution times [3], optimal control, when the applications are controllers [4], and when dealing with energy minimization [5].

In our previous work [16], we have proposed a number of control algorithms to deal with execution time variations in a uniprocessor system.

Lu et al. [6] described a framework for feedback control scheduling, where the source of nondeterminism is execution time variation, and the actuation method is admission control and task rate change.

Palipoli et al. [7], [8] proposed a feedback based technique for adjusting the parameters of a resource reservation scheduling algorithm in reaction to task execution time variations.

Cervin et al. [13], [14] proposed a method for adjusting control performance of tasks that implement feedback controllers.

Combaz et al. [9] proposed a QoS scheme for applications composed of tasks, each described as graphs of subtasks, and each subtask has a number of modes.

Liu et al. [10], [11] presented a Recursive Least Squares based controller to control the utilization of a distributed system by means of rate adjustment.

Ad-hoc approaches suffer due to their lack of formal guarantees. Control theory based approaches potentially address this issue, but the results they offer are typically restrictive, due to the simplicity of the models used. In most cases, the models are coarse-grain, linear models, where variations in load are modeled as Poisson processes or white noise [6], [12]. Stability is only shown, if at all, under very particular and restrictive assumptions.

In this work, we overcome the above limitations. We consider a real-time system running a set of independent tasks and resource managers that control the processor's utilization. Our aim is to develope general models of the system and determine conditions that a resource manager must meet in order to render it stable. For the developed system models, we also determine bounds on the worst case

response times of tasks.

II. Preliminaries

A. System and Application

We consider a uniprocessor system running a set of independent tasks (Λ):

$$\Lambda = \{\tau_i, i \in I\}$$

where τ_i is a task and I is a finite index set. A task in the system is defined as a tuple:

$$\tau_i = (\mathbf{P}_i = [\rho_i^{\min}, \rho_i^{\max}], \mathbf{C}_i = [c_i^{\min}, c_i^{\max}])$$

where $\mathbf{P}_i$ is the interval of possible job release rates for the task and $\mathbf{C}_i$ is the interval of possible execution times for jobs of this task. The response time of any job of a task represents the interval of time between the release and the finish time of the job. We denote by $\mathbf{P} = \prod_{i \in I} \mathbf{P}_i$ and $\mathbf{C} = \prod_{i \in I} \mathbf{C}_i$ the spaces of rates and execution times for the tasks in Λ, and with $\overline{\rho}$ and $\overline{c}$ points in this spaces. A job of a task in the system is defined as:

$$\tau_{ij} = \left(c_{ij}, \rho_{ij}, r_{ij}\right)$$

where: c_{ij}, ρ_{ij}, and r_{ij}, are the execution time, rate, and response time of the jth job of task i.

The tasks in the system are scheduled using any scheduler which has the following properties:

1) it is *non-idling*: it does not leave the processor idle if there are pending jobs;
2) it executes successive jobs of the same task in the order of their release time.

A resource manager is running on the processor, whose goal is to measure execution times, and then adjust job release rates for all tasks, such that the CPU utilization is kept high, and the amount of time spent in overload situations is minimized. We consider the system stable if under the worst possible run-time scenario, the overload in the system is kept finite, meaning that the system does not keep accumulating jobs without having a way of executing them.

B. Processor Behavior in Overload Situations

In overload situations jobs cannot be executed at the rate at which they are released, because of the larger than expected job execution times. In this condition, newly released jobs need to wait for the previous ones to finish. We consider that newly released jobs queue up in queues, one for each task. A job τ_{ij} gets executed only after all previous queued up jobs of task τ_i finished executing.

C. Resource Utilization and Schedulability

The resource considered in this paper is the CPU time. The *resource utilization*, in any interval of time h, is the fraction of h when the CPU is non-idle. This is a positive number less/equal 1:

$$u = \frac{\text{non-idle time}}{h}, u \in [0, 1]$$

The *resource demand*, in an interval of time h, is:

$$u^D = \frac{C_{\text{previous}} + C_{\text{current}}}{h}, u^D \geq 0$$

where C_{current} is the sum of execution times of all jobs released during h and C_{previous} is the sum of execution times of all queued up jobs, released before the beginning of the interval. The resource demand may be larger then 1. Figure 1 shows an example with three tasks. At time instance $t_{[k-1]}$, task τ_1 has 2 jobs in its queue, τ_2 has 4, and τ_3 has 1. C_{previous} will be the sum of the execution times of all these jobs. Between $t_{[k-1]}$ and $t_{[k]}$, τ_1 releases 3 new jobs, τ_2 releases 2, and τ_3 1. In this case C_{current} will be equal to the sum of execution times of all these six jobs.

There are two observations to be made: 1) at $t_{[k-1]}$ some of the jobs in the queues will be partially executed; in this case we consider only their remaining execution time in C_{previous}[1]; 2) jobs are meant to finish by the end of their period[2], not by the end of the time interval h; therefore, when a job has its end of the period after $t_{[k]}$, we will only consider the part of its execution time, corresponding to the portion of its period inside the interval of time h, as part of C_{current}.

When the resource demand is less or equal to 1, then it will be equal with the resource utilization and the system will be schedulable; all the demand can be executed. When the resource demand is above 1 the system is overloaded, only a portion of the demand can be executed (namely 1), and jobs start accumulating in queues. Also, we must note that execution times change with every job, and they are unknown before the job's completion. Therefore, at any moment, the resource demand can only be predicted.

We consider that the employed resource manager controls the system by adjusting job release rates and, thus, controlling the resource demand.

III. Problem Formulation

We consider any task set Λ and a resource manager whose job is to keep the resource demand $u^D = 1$ by adjusting task rates. Our goal is to model the resource demand, the behavior of the real-time system and resource manager in order to determine conditions under which the whole system is stable. By stability we mean that the resource demand in the system is bounded and job response times do not grow infinitely.

IV. System Modeling

In loose terms, the model of our system can be depicted as in Figure 2. While the tasks and the resource manager run on the same CPU, from the modeling perspective the tasks form the plant, and the resource manager is the controller controlling the resource demand of the plant by means of adjusting task rates.

[1] if a queue has 5 jobs in it, and one was half executed, then we consider the sum of execution times for the remaining 4.5 jobs

[2] a jobs period is the inverse of its rate.

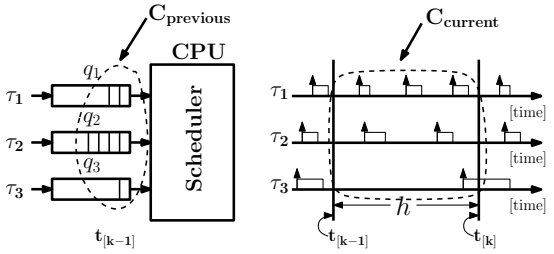

Figure 1. Resource demand in the system, during time interval h.

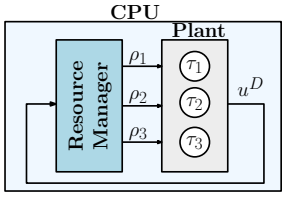

Figure 2. A system is seen as a control loop, where the resource manager controls the resource demand by adjusting task rates. The tasks constitute the controlled plant.

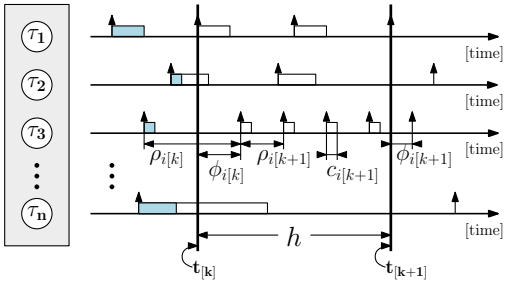

Figure 3. Parameters related to the resource demand, and the state of the system.

A. Modeling of the Real-Time System

When modeling a system, the goal is to know at each moment in time $t_{[k]}$ what will the system's state be, in the future. This is a discrete-time dynamical system where the control is done at discrete moments in time $t_{[k]}$. We model our real-time system (the ensemble of resource manager and plant) as a system of difference equations:

$$F(\overline{x}_{[k+1]}, \overline{x}_{[k]}, \cdots) = 0 \tag{1}$$

where $\overline{x}_{[k+1]}$ and $\overline{x}_{[k]}$ are the state vectors of the system at the future ($t_{[k+1]}$) and the current ($t_{[k]}$) moments of time, and F is some function of the state vectors and possibly other parameters (e.g. noise). In our real-time system, the state is represented by the resource demand u^D and, therefore, the state vector will be comprised of the elements that determine it: queue sizes, task rates, and execution times.

Figure 3 presents a system with n tasks, and shows how its state evolves between two successive actuations of the resource manager (controller). In order to control the resource demand during the time interval $[t_{[k]}, t_{[k+1]}]$, at time instance $t_{[k]}$, the controller will choose new task rates $\rho_{i[k+1]}$, based on queue sizes ($q_{i[k]}$ – not shown in this figure) and future average job execution times ($c_{i[k+1]}$). The future average execution times are unknown and must be predicted based on previously finished jobs.

There are several observations to be made. It can happen that some tasks (such as τ_1) release jobs precisely at the time instance at which the controller runs, and have periods (inverse of the rates) that are multiple of the time interval between the two actuations. However, in general, this is not the case and new jobs will be released with an offset ($\phi_{i[k]}$) from the start of the time interval ($t_{[k]}$) and do not typically end at the end of the time interval. Also there may be tasks (such as τ_n) which will not release any job during this interval.

In Sections V and VI we will develope concrete models of our system.

B. Stability of Discrete-Time Dynamical Systems

A discrete-time dynamical system is a tuple $\{\mathcal{T}, \mathcal{X}, \mathcal{A}, \mathcal{S}\}$ where $\mathcal{T} = \{t_{[k]} | k \in \mathbb{N}, t_{[k]} > t_{[k-1]} > \cdots > t_{[0]} = 0\}$ is the set of actuation times, $\mathcal{X}$ is the state space, $\mathcal{A} \subset \mathcal{X}$ is the set of all initial states of the system ($\overline{x}_{[0]}$), and $\mathcal{S}$ is the set of all trajectories (all solutions of (1) : $\overline{x}_{[k]} = p(t_{[k]}, \overline{x}_{[0]})$ where $t_{[k]} \in \mathcal{T}$ and $\overline{x}_{[0]} \in \mathcal{A}$). Also the state space must be a metric space $(\mathcal{X}, d)$, where $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is a distance function between two state vectors in $\mathcal{X}$. We use the notation $d(\overline{x}, \mathcal{M}) = \inf_{\overline{y} \in \mathcal{M}} \{d(\overline{x}, \overline{y})\}$ to describe the distance from a state to a set $\mathcal{M}$.

A dynamical system's state is desired to belong to a preselected bounded set of points $\mathcal{M} \subset \mathcal{A}$. Because systems are typically subject to noises, this condition does not hold. Under this premises a dynamical system is said to be stable if its state remains "close" to $\mathcal{M}$ under all noise patterns and initial conditions.

For our real-time system, we will consider the notion of *stability in the sense of Lagrange* [17], where a system is stable if all trajectories starting from states within a ball of size δ around $\mathcal{M}$ are bounded in a larger ball of size $\gamma(\delta)$ around $\mathcal{M}$ ($\overline{x}_0 \in B(\delta) \Rightarrow p(t_{[k]}, \overline{x}_{[0]}) \leq B(\gamma(\delta))$). To test for stability, we shall use the following theorem described in [17]:

Theorem 1 (uniform boundedness): Let us consider the above described dynamical system. Assume that there exist a function $V : \mathcal{X} \to \mathbb{R}_+$ and two strictly increasing functions $\varphi_1, \varphi_2 : \mathbb{R}_+ \to \mathbb{R}_+$ with $\lim_{r \to \infty} \varphi_i(r) = \infty, i = \overline{1,2}$, such that

$$\varphi_1(d(\overline{x}, \mathcal{M})) \leq V(\overline{x}) \leq \varphi_2(d(\overline{x}, \mathcal{M})) \tag{2}$$

for all $\overline{x} \in \mathcal{X}$ whenever $d(\overline{x}, \mathcal{M}) \geq \Omega$, where Ω is a positive constant.

Also, assume that $V(p(t_{[k]}, \overline{x}_{[0]}))$ is non-increasing for all trajectories $p(t_{[k]}, \overline{x}_{[0]}) \in \mathcal{S}$ whenever $d(p(t_{[k]}, \overline{x}_{[0]}), \mathcal{M}) \geq \Omega$. Assume that there exists a constant $\Psi > 0$ such that $d(p(t_{[k+1]}, \overline{x}_{[0]}), \mathcal{M}) \leq \Psi$ whenever $d(p(t_{[k]}, \overline{x}_{[0]}), \mathcal{M}) \leq \Omega$.

If the above assumptions hold, then the system is *uniformly bounded.* ◇

Any system that satisfies the above theorem is stable, and this means that its state becomes trapped in the ball of size $\max\{\Omega, \Psi\}$ around the set $\mathcal{M}$, after a certain amount of time (possibly infinite), regardless of the initial state $\overline{x}_{[0]}$. In practice however, only initial states within Ω will be of relevance to us, making $\max\{\Omega, \Psi\}$ a performance metric for our system.

V. Constrained Model of the Real-Time System

In this section we develop a simple but constrained model for our system and determine conditions under which it is stable. In Section VI we will derive a generalized model of the system.

A. Assumptions

The constrained model is based on the assumption that the time interval between two successive actuations of the controller (the controller's period) $h = t_{[k+1]} - t_{[k]}$ is much larger then the largest possible offset in the system:

$$\phi_{\max} < \max_{i \in I} \Big\{ \frac{1}{\rho_i^{\min}} \Big\} \ll h$$

Since h is large, we can neglect offsets in this model. Systems described by this model will be called *constrained systems*.

B. Model

We will start to model our systems from the definition of resource demand (Section II-C). By disregarding task offsets, the formula is:

$$u^D_{[k]} = \frac{1}{h} \Big(\underbrace{\sum_{i \in I} c_{i[k]} \cdot q_{i[k-1]}}_{C_{\text{previous}}} + \underbrace{\sum_{i \in I} c_{i[k]} \cdot \rho_{i[k]} \cdot h}_{C_{\text{current}}} \Big) \tag{3}$$

The first sum represents the accumulation of execution times from previously released, but not executed (queued up) jobs. The second sum represents the accumulation of execution times from jobs that are released during $[t_{[k-1]}, t_{[k]}]$. We note that $c_{i[k]}$ represents the average execution time of the jobs of τ_i that were executed during $[t_{[k-1]}, t_{[k]}]$. The resource demand $u^D_{[k]}$ corresponds to an amount of execution time, that needs to be executed by the end of the period. If this amount is less than h, then it will be executed entirely, otherwise only h out of it will get executed.

As we can observe from Equation (3), the model of our system must contain states for the queue sizes, execution times, and rates. We will model the evolution of the queue sizes implicitly, by considering them together, in terms of an amount of execution time. We will then have:

$$\sum_{i \in I} c_{i[k+1]} \cdot q_{i[k+1]} = h \cdot \max \Big\{ 0, u^D_{[k+1]} - 1 \Big\} \tag{4a}$$

$$\overline{q^{\text{p}}}_{[k+1]} = \overline{q}_{[k]} \tag{4b}$$

We use the notation $\overline{q}_{[k]}$ for the vector of queue sizes, and $\overline{q^{\text{p}}}_{[k]}$ for the vector of queue sizes at the previous time instance. Equation (4b) is needed because we wish to make the resource demand a function of the state of the system $u^D_{[k]} \overset{not}{=} u^D(\overline{x}_{[k]})$. Observe that queue sizes are modeled as being continuous values. This is because the jobs at the top of the queues are partially executed and in our model we account for that (see Section II-C).

Next, we model the average execution time of the jobs that will be executed in the current interval of time $[t_{[k]}, t_{[k+1]}]$, as being some function $f_p : \mathbf{C} \to \mathbf{C}$ of the average execution time in the previous time interval plus some noise $\overline{\nu}$. The function f_p is then a prediction function, and $\overline{\nu}$ is a random variable which is unknown:

$$\overline{c}_{[k+1]} = f_p(\overline{c}_{[k]}) + \overline{\nu}_{[k]} \tag{4c}$$

Observe that, irrespective of the noise, $\overline{c}_{[k+1]} \in \mathbf{C}$ holds. Equations (4a) to (4c) represent the model of the plant. We complete the model of the system with the model of the controller. The controller's job is to compute new rates:

$$\overline{\rho}_{[k+1]} = f_c(\overline{x}_{[k]}) \tag{5}$$

Here we only model the controller in a general way, as being any function of the state of the system. The state vector of the system is:

$$\overline{x}_{[k]} = \Big(\overline{q}_{[k]}, \overline{q^{\text{p}}}_{[k]}, \overline{c}_{[k]}, \overline{\rho}_{[k]} \Big)^T.$$

Equations (4a) to (4c), and (5) represent the model of our real-time system. Our model is a model of a discrete-time dynamical system $\{\mathcal{T}, \mathcal{X}, \mathcal{A}, \mathcal{S}\}$ where $\mathcal{T}$ is defined as in Section IV-B, $\mathcal{X} = \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbf{C} \times \mathbf{P}$, $\mathcal{A} = \mathcal{X}$ and S is the set of all solutions of equations (4a) to (4c), and (5). We make the observation that $\mathcal{X} \subset \mathbb{R}^{4n}$ where n is the number of tasks in the system and, therefore, is a metric space for the usual distances. Throughout the rest of this paper we will use the Chebyshev distance [18]:

$$d(\overline{x}, \overline{y}) = \sup_{i = \overline{1,4n}} \{|x_i - y_i|\}$$

The above model describes a class of systems, namely the systems generated by all combinations of existing f_p and f_c functions, and all allowable schedulers (see Section II), for all instances of applications Λ. *Our goal is to determine the subclass of controllers that leads to stable systems.*

C. Stability

In this section we want to determine conditions that our controller must satisfy, in order for the system to be stable (satisfy Theorem 1). We first define the notions involved in Theorem 1 and then we determine the stability conditions.

We define the set of states $\mathcal{M}$ as being all the states where $u^D = 1$. The following lemma determines that $\mathcal{M}$ is bounded.

Lemma 1: The set $\mathfrak{M}^\alpha = \{\overline{x} \in \mathcal{X} | u^D(\overline{x}) = \alpha\}$ is bounded, where $\alpha > 0$ is an arbitrary constant. $\diamond$

Proof: From equation (4a), (4b) and (3) we have

$$\sum_{i \in I} c_i \cdot q_i^{\text{p}} = h\big(\alpha - \sum_{i \in I} c_i \cdot \rho_i\big)$$

and it quickly follows that the largest distance between two states in $\mathfrak{M}^\alpha$ is bounded by

$$d^\alpha = \max_{j \in I} \{|q_j^\alpha - 0|, |q_j^{\text{p}\alpha} - 0|, |c_j^{\max} - c_j^{\min}|, |\rho_j^{\max} - \rho_j^{\min}|\}$$

$$\varphi_1(d) = \begin{cases} \frac{\min_{j\in I}\{c_j^{\min}\}}{h} \cdot d + \sum_{i\in I} c_i^{\min} \cdot \rho_i^{\min} - 1, & \text{if } d \geq \Omega > d^1 \\ \Big(\frac{\min_{j\in I}\{c_j^{\min}\}}{h} + \frac{1}{\Omega}\big(\sum_{i\in I} c_i^{\min} \cdot \rho_i^{\min} - 1\big)\Big) \cdot d, & \text{otherwise} \end{cases} \tag{6}$$

$$V(\overline{x}) = \max\Big\{0, \sum_{i\in I} c_i^{\max}\big(\frac{q_i^{\mathrm{p}} + dq_i^{\mathrm{p}} + 1}{h} + \rho_i + d\rho_i\big) - 1\Big\} \leq \frac{\alpha_d}{h}\sum_{i\in I} c_i^{\max} + \max\Big\{0, \sum_{i\in I} c_i^{\max}\big(\frac{q_i^{\mathrm{p}\alpha_d} + 1}{h} + \rho_i^{\max}\big) - 1\Big\}, \forall\, \overline{x} \in \mathcal{M}_{\alpha_d} \tag{7}$$

where:

$$q_j^{\mathrm{p}\alpha} = \frac{h}{c_j^{\min}}\Big(\alpha - \sum_{i\in I} c_i^{\min} \cdot \rho_i^{\min}\Big),$$

and

$$q_j^{\alpha} = \frac{h}{c_j^{\min}} \cdot \max\Big\{0, \alpha - 1\Big\}, \qquad \forall j \in I \qquad \blacksquare$$

Since $\mathcal{M} = \mathfrak{M}^1$, $\mathcal{M}$ is bounded as well.

The following theorem gives a necessary condition for a system to be stable.

Theorem 2: A necessary condition for a system to be stable is:

$$\sum_{i\in I} c_i^{\max} \cdot \rho_i^{\min} \leq 1 \tag{8}$$
$\diamond$

Proof: We prove this by contradiction. We allow Equation (8) not to hold and we will show that this leads to an unstable system. If equation (8) does not hold, then we have:

$$\sum_{i\in I} c_i^{\max} \cdot \rho_i^{\min} - 1 = \beta > 0$$

In the case that $c_{i[k]} = c_i^{\max}$, even if the controller sets the smallest rates ($\overline{\rho}_{[k]} = \overline{\rho}^{\min}$), for all $k \geq 0$, we have from above, and from equations (3), and (4a) that:

$$u_{[k+1]}^D - 1 = u_{[k]}^D - 1 + \beta \quad \Rightarrow \quad u_{[k+1]}^D = u_{[0]}^D + (k+1)\cdot\beta$$

and thus $\lim_{k\to\infty} u_{[k]}^D = \infty$. $\blacksquare$

Theorem 2 implies that, in order to achieve stability, there must exist at least a rate vector (the rate vector consisting of the smallest rates for each task), which the controller can choose such that, when jobs have worst case execution times, the contribution to resource demand in each time interval is not more than 1. Otherwise, the queue sizes will continue growing irrespective of the controller and scheduler used, and the resource demand will be unbounded. For the rest of the paper, we will only consider systems which satisfy Theorem 2. We continue by defining the set:

$$\Gamma_\star = \Big\{\overline{\rho}_* \in \mathbf{P} \;\Big|\; \sum_{i\in I} c_i^{\max} \cdot \rho_{i*} \leq 1\Big\} \neq \emptyset \tag{9}$$

which is the set of all rate vectors that, if chosen, guarantee that job queue sizes do not continue growing, irrespective of the execution times of the jobs. If the system satisfies Theorem 2, $\Gamma_\star$ contains at least one rate vector.

Next, we determine sufficient conditions for the controller, in order to render the system stable.

Theorem 3: For any constrained system that satisfies Theorem 2 and for any $u_\Omega^D > 1$ a sufficient stability condition is that its controller satisfies:

$$\overline{\rho}_{[k+1]} \in \begin{cases} \Gamma_\star, & \text{if } u_{[k]}^D \geq u_\Omega^D \\ \mathbf{P}, & \text{otherwise} \end{cases} \tag{10}$$
$\diamond$

Proof: Ultimately, stability means that queue sizes ($\overline{q}$ and $\overline{q^{\mathrm{p}}}$) have upper bounds. We first proceed on defining the function $V(\overline{x}_{[k]})$ (noted $V_{[k]}$ henceforward) and finding the two function $\varphi_1(d(\overline{x}_{[k]}, \mathcal{M}))$ and $\varphi_2(d(\overline{x}_{[k]}, \mathcal{M}))$ such that equation (2) holds. By inspiring ourselves from the model of queue sizes (equations (16a) and (16d)), and considering the worst-case noises in the system (noises are due to inaccuracies in predicting execution times and in measuring queues), we define $V_{[k]}$ as:

$$V_{[k]} = \max\Big\{0, \frac{1}{h}\sum_{i\in I} c_i^{\max}\cdot(q_{i[k]}^{\mathrm{p}} + 1) + \sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]} - 1\Big\} \tag{11}$$

To determine φ_1 we observe that $V_{[k]} \geq u_{[k]}^D - 1$. We then construct the set

$$\mathcal{V}_{\alpha_V} = \big\{\overline{x} \in \mathcal{X}\big|u^D(\overline{x}) - 1 = \alpha_V\big\} = \mathfrak{M}^{1+\alpha_V}$$

where $\alpha_V > 0$ is an arbitrary constant. We observe that a bound on the largest distance between a state in $\mathcal{V}_{\alpha_V}$ and a state in $\mathcal{M}$ is given by $d^{1+\alpha_V}$, and that:

$$d^{1+\alpha_V} = \frac{h}{\min_{j\in I}\{c_j^{\min}\}}\Big(\alpha_V + 1 - \sum_{i\in I} c_i^{\min}\cdot\rho_i^{\min}\Big)$$

for all $d^{1+\alpha_V} \geq d^1$. We can also say that $d(\overline{x}_{[k]}, \mathcal{M}) \leq d^{1+\alpha_V}$ and $V(\overline{x}) \geq \alpha_V$, $\forall \overline{x} \in \mathcal{V}_{\alpha_V}$. Since α_V was chosen arbitrarily and the above expression of $d^{1+\alpha_V}$, as a function of α_V, is invertible, we obtain $\varphi_1(d(\overline{x}_{[k]}, \mathcal{M}))$ in equation (6) which satisfies the conditions in Theorem 1.

To determine φ_2 we construct the set of all states at a distance α_d or less from $\mathcal{M}$:

$$\mathcal{M}_{\alpha_d} = \{\overline{x} \in \mathcal{X} | d(\overline{x}, \mathcal{M}) \leq \alpha_d\} = \{\overline{x} \in \mathcal{X} | d(\overline{x}, \overline{y}) = \alpha_d,\ \forall \overline{y} \in \mathcal{M}\}$$

where $\alpha_d > 0$ is an arbitrary constant. We than determine which of the states in $\mathcal{M}_{\alpha_d}$ has the highest value of V. If

$$\overline{y} = \Big(\overline{q}, \overline{q^{\mathrm{p}}}, \overline{c}, \overline{\rho}\Big)^T \in \mathcal{M} \text{ and}$$

$$\overline{x} = \Big(\overline{q} + d\overline{q}, \overline{q^{\mathrm{p}}} + d\overline{q^{\mathrm{p}}}, \overline{c} + d\overline{c}, \overline{\rho} + d\overline{\rho}\Big)^T \in \mathcal{M}_d$$

then we have inequality (7) and, since α_d was chosen arbitrarily, we obtain $\varphi_2(d(\overline{x}, \mathcal{M}))$ as being the right half of the inequality (7).

$$V_{[k+1]} \leq \sum_{i\in I} c_i^{\max}\frac{q_{i[k]}+1}{h} = \sum_{i\in I} c_i^{\max}\big(\frac{q^{\mathrm{p}}_{i[k]}+1}{h}+\rho_{i[k]}\big) - \frac{1}{h}\sum_{i\in I} c_i^{\max}\cdot e_{i[k]} \leq V_{[k]} \tag{12}$$

To determine Ω, we construct the set $\mathcal{V}_{u_\Omega^D-1}$ and we observe that $\Omega = d^{u_\Omega^D}$. It seems that we can only choose u_Ω^D sufficiently large such that $d^{u_\Omega^D} \geq d^1$, however in practice, we can build an equivalent model of the system by applying a linear transformation to the state space, where this condition always holds, thus this is a non-issue.

Next, we proceed on showing that $V_{[k+1]} \leq V_{[k]}$, $\forall d(\overline{x}_{[k]}, \mathcal{M}) \geq \Omega$. In this case, we have from equations (4a), (4b), (10), and (11):

$$V_{[k+1]} = \max\left\{0, \sum_{i\in I} c_i^{\max}\frac{q_{i[k]}+1}{h} + \underbrace{\sum_{i\in I} c_i^{\max}\cdot\rho_{i[k+1]} - 1}_{\leq 0}\right\}$$

When the value inside the max function is larger than 0, we observe from equations (4a) and (3) that $q_{i[k]} = q^{\mathrm{p}}_{i[k]} + \rho_{i[k]}\cdot h - e_{i[k]}$, where $e_{i[k]} \geq 0$ is the number of jobs of τ_i executed during the time interval $[t_{[k-1]}, t_{[k]}]$, and $\frac{1}{h}\sum_{i\in I} c_{i[k]}\cdot e_{i[k]} = 1$. $e_{i[k]}$, for all $i \in I$ depend, amongst others, on the scheduler used, and are typically unknown. Regardless of their value, however, inequation (12) holds.

To complete the proof, we must show that there exists a value $\Psi > 0$ such that for all states with $d(\overline{x}_{[k]}, \mathcal{M}) \leq \Omega$ we have $d(\overline{x}_{[k+1]}, \mathcal{M}) \leq \Psi$. Since $q_{i[k]} \leq \Omega$, $c_{i[k]} \leq c_i^{\max}$, and $\rho_{i[k]} \leq \rho_i^{\max}$, $\forall i \in I$, we then have

$$u^D_{[k+1]} \leq \sum_{i\in I} c_i^{\max}\big(\frac{\Omega+1}{h} + \rho_i^{\max}\big) = u^D_\Psi \tag{13}$$

All subsequencent states $\overline{x}_{[k']}$, $k' \geq k+1$ will have $u^D_{[k']} \leq V_{[k+1]} + 1$ if $u^D_\Psi \geq u^D_\Omega$ or, otherwise $u^D_{[k']} \leq u^D_\Omega$. Then we have $\Psi = \varphi_2^{-1}(\max\{u^D_\Psi, u^D_\Omega\})$.

With the above, the proof of Theorem 3 is complete. ∎

Any system that satisfies the above theorem is stable. Observe that the condition (10) is a condition for the controller used in the system. As long as the task execution times and rates are such that $u^D_{[k]} < u^D_\Omega$, the controller is free to chose any rate $\rho_{i[k+1]} \in [\rho_i^{\min}, \rho_i^{\max}]$. The controller will choose rates according to the particular control goals requested by the application (e.g. keeping processor utilization high, providing QoS guarantees, etc.). Once $u^D_{[k]} \geq u^D_\Omega$ (which means that the system reached a certain critical level of overload), the controller will have to choose rate vectors from the set $\Gamma_\star$, and keep doing so until a time instance $t_{[k']}$, when $u^D_{[k']} < u^D_\Omega$.

VI. General Model of the Real-Time System

In this section we extend the previous model of a real-time system, by considering task offsets. This will allow us to consider controller periods which are arbitrary, therefore removing the limitation of the previous model (Section V). Systems described according to this model will be called *general systems*.

This section is organized as the previous one: we start by giving a model of the system and, then, we analyze its stability.

A. Model

We first describe the *resource request*:

$$u^R_{[k]} = \sum_{i\in I} c_{i[k]}\cdot\frac{q^{\mathrm{p}}_{i[k]} + \lceil\rho_{i[k]}\cdot\max\{0, h-\phi_{i[k-1]}\}\rceil}{h} \tag{14}$$

According with the definition in Section II-C, the resource demand is:

$$u^D_{[k]} = u^R_{[k]} - \frac{1}{h}\cdot\sum_{i\in I} c_{i[k]}\cdot\rho_{i[k]}\cdot\phi_{i[k]} \tag{15}$$

The resource request $u^R_{[k]}$ represents the amount of execution times of the queued-up jobs and jobs released during h, while $u^D_{[k]}$ represents only the part of $u^R_{[k]}$ that should actually be executed during h. For the constrained model in Section V, $u^D_{[k]} = u^R_{[k]}$, since we ignore offsets. The following equation models the behavior of the queues:

$$\sum_{i\in I} c_{i[k+1]}\cdot q_{i[k+1]} = h\cdot\max\{0, u^R_{[k+1]} - 1\} \tag{16a}$$

This is slightly different from Section V-B because in this case the accumulation of execution time is given by $h\cdot u^R_{[k]}$. The most notable difference compared to the constrained model, however, is the addition of offsets to the state:

$$\phi_{i[k+1]} = \phi_{i[k]} + \frac{1}{\rho_{i[k+1]}}\lceil\rho_{i[k+1]}\cdot\max\{0, h-\phi_{i[k]}\}\rceil - h \tag{16b}$$

for each task in the system. The rest of the model is:

$$\overline{c}_{[k+1]} = f_p(c_{i[k]}) + \nu_{[k]} \tag{16c}$$

$$\overline{q^{\mathrm{p}}}_{[k+1]} = \overline{q}_{[k]} \tag{16d}$$

$$\overline{\phi^{\mathrm{p}}}_{[k+1]} = \overline{\phi}_{[k]} \tag{16e}$$

Equations (16a) to (16e) represent the model of the plant. This model has states similar to the constrained model presented in Section V-B, but is augmented with new states corresponding to offsets (equations (16b)). Equations (16d) and (16e) are part of the model because we want to make the resource demand a function of the state, so that we can control it. We assume the same model for the controller as in Equation (5). The general model of a real-time system is, thus, given by equations (16a) to (16e) and (5).

Since the number of states has grown, the state space is larger but is again of the form $\mathbb{R}^n$ and we shall again use the Chebyshev norm. Also note that offsets are bounded by $\phi_i \in [0, 1/\rho_i^{\max}]$, $\forall i \in I$.

$$V_{[k]} = \begin{cases} \max\left\{0, \sum_{i\in I} c_i^{\max} \frac{q^{\mathrm{p}}_{i[k]}+1+\left\lceil \rho_{i[k]}\cdot\max\{0,h-\phi^{\mathrm{p}}_{i[k]}\}\right\rceil+1-\rho_{i[k]}\cdot\phi_{i[k]}}{h} - 1\right\} & , \text{ if } \overline{\rho}_{[k]} \in \Gamma_\star \\ \max\left\{0, \sum_{i\in I} c_i^{\max} \frac{q^{\mathrm{p}}_{i[k]}+1+\left\lceil \rho_{i[k]}\cdot\max\{0,h-\phi^{\mathrm{p}}_{i[k]}\}\right\rceil+1-\rho_{i[k]}\cdot\phi_{i[k]}}{h} + \sum_{i\in I} c_i^{\max} - 1\right\} & , \text{ otherwise.} \end{cases}$$

$$= \begin{cases} \max\left\{0, \frac{1}{h}\sum_{i\in I} c_i^{\max}\left(q^{\mathrm{p}}_{i[k]}+2\right) + \sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]} - 1 - \frac{1}{h}\sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]}\cdot\phi^{\mathrm{p}}_{i[k]}\right\} & , \text{ if } \overline{\rho}_{[k]} \in \Gamma_\star \\ \max\left\{0, \frac{1}{h}\sum_{i\in I} c_i^{\max}\left(q^{\mathrm{p}}_{i[k]}+3\right) + \sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]} - 1 - \frac{1}{h}\sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]}\cdot\phi^{\mathrm{p}}_{i[k]}\right\} & , \text{ otherwise.} \end{cases} \quad (17)$$

(18)

B. Stability

Theorem 4: For any general system that satisfies Theorem 2 and for any $u^D_\Omega > 1$, a sufficient stability condition is that its controller satisfies:

$$\overline{\rho}_{[k+1]} \in \begin{cases} \Gamma_\star, & \text{if } u^D_{[k]} \geq u^D_\Omega \\ \mathbf{P}, & \text{otherwise} \end{cases} \quad (19)$$

$$\rho_{i[k+1]} \leq \rho_{i[k]} \quad \text{if } u^D_{[k]} \geq u^D_\Omega, \quad \forall i \in I \quad (20)$$

◇

Proof: Compared with the constrained model, we have and extra condition that the controller must satisfy (equation (20)). We want to show that the conditions are sufficient to make $\overline{q}_{[k]}$ and $\overline{q^{\mathrm{p}}}_{[k]}$ bounded.

We define the function V as in equation (17). At any moment of time $t_{[k]}$, the V function represents the accumulation of execution times, that should have been executed by $t_{[k]}$[3]. Since in each queue, we consider the whole execution time of all released jobs, we must remove the part of each job that should be executed after the end of the resource manager's period (this done by the term $-\frac{1}{h}\sum_{i\in I} c_i^{\max}\cdot\rho_{i[k]}\cdot\phi^{\mathrm{p}}_{i[k]}$).

In Figure 4 we consider a system of two tasks, and we consider that $u^D_{[k']} \geq u^D_\Omega$, $\forall k' \geq k$. At each moment in time $t_{[k]}$ new rates are computed, but jobs are released with this rates only after a certain offset (marked by the arrows in the figure). From condition (19) we are certain that during the intervals of time T_3, T_5, T_7, T_9, ...the accumulation of execution times added in the system is less than the size of the respective intervals, therefore the total accumulation drops. Condition (20) ensures that the same happens over the transition intervals $T_4, T_6, T_8, T_{10}, \ldots$ The accumulation of execution times from the intervals T_1 and T_2 may be larger than $T_1 + T_2$ because $\overline{\rho}_{[k]} \notin \Gamma_\star$, however, it is bounded by $\sum_{i\in I} c_i^{\max}$. This explains the two cases in the definition of V. The other constants that appear are there so that $u^D_{[k]} - 1 \leq u^R_{[k]} - 1 \leq V_{[k]}$ holds. Otherwise, since the V function is very similar to the one defined in equation (11), all the steps of this proof are very similar with the proof of Theorem 3 and in the interest of brevity we shall not discuss them here.

[3]Strictly speaking, V represents a load value. To obtain the accumulation of execution times, one must multiply the function with h.

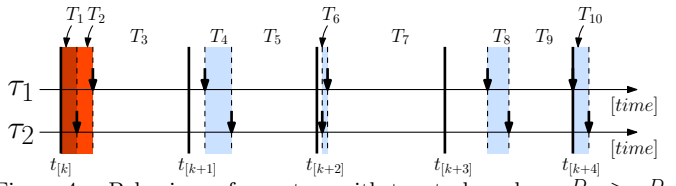

Figure 4. Behaviour of a system with two tasks, when $u^D_{[k']} \geq u^D_\Omega$, $\forall k' \geq k$.

For determining Ψ, we use a similar reasoning as in Theorem 3 and we obtain that:

$$u^D_{[k+1]} \leq u^R_{[k+1]} \leq \sum_{i\in I} c_i^{\max}\left(\frac{\Omega+1}{h} + \rho_i^{\max}\right) = u^D_\Psi \quad (21)$$

Thus, any general system satisfying the conditions in Theorem 4 is stable. ■

We observe that for the general model, the stability criterion is more restrictive than for the constrained model. This is because the general model considers in more depth the transition between different rates, and more conditions are needed to ensure that this transition is done safely. Also, for both models, we have required that $V(\overline{x}_{k+1}) \leq V(\overline{x}_k)$ holds. This means that if the starting point of the system is outside Ω, the system may never reach Ω within a finite amount of time. This is not a problem in practice, because systems usually start with empty queues, thus from within Ω. However, the $\Gamma_\star$ can easily be modified such that stronger assumption for V hold.

VII. Worst Case Response Time Bound

For any controller that satisfies the stability condition in Theorem 4 there exists a finite response time for each task. The actual value of the response time depends on the concrete scheduling policy and the controller (f_c) used in the system. In this paper we will develop bounds on the worst case response time for tasks considering an EDF scheduler [15] and two classes of controllers. The bounds developed here are different from the well known worst case response times derived in literature for EDF, since our system allows overload situations. The EDF scheduler considers as a working deadline for each job τ_{ij}, the sum of its release time and $1/\rho_{ij}$, where ρ_{ij} is the current job's rate.

For the following analysis, we will consider that the system always starts from a state where the queues are

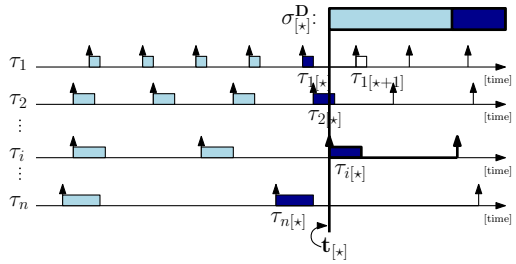

Figure 5. Accumulation of execution times

empty.

$$\mathcal{A} = \left\{ \overline{x}_{[0]} \in \mathcal{X} \middle| q = 0 \right\} \tag{22}$$

In this case $d(\overline{x}_{[0]}, \mathcal{M}) \leq d^1 \leq \Omega$, $\forall \overline{x} \in \mathcal{A}$ holds. According to equations (13) and (21) (for the constrained and general model), u^D_Ψ is the highest resource demand (and resource request) ever achieved in the system. This result is important, since it allows us to bound the overload in the system.

At a certain moment in time $t_{[\star]}$, a new job of a task τ_j is released and we wish to compute its response time. We will denote this job with $\tau_{j[\star]}$. In the system, at $t_{[\star]}$ there already exists a certain number of un-executed jobs and their total accumulation of execution times is:

$$\sigma^D_{[\star]} = \sum_{i \in I} c_{i[\star]} \cdot q_{i[\star]}$$

where $q_{i[\star]}$, $\forall i \in I$ are the queue sizes of each task and $c_{i[\star]}$ are the average execution time for the jobs in the queues (these averages are unknown, but $\overline{c}_{[\star]} \in \mathbf{C}$). Figure 5 illustrates this situation for a system of n tasks. All the jobs depicted in the figure are not executed at the moment $t_{[\star]}$, when $\tau_{j[\star]}$ is released. The dark shaded jobs represent the last released jobs of the tasks, before the moment $t_{[\star]}$. The light shaded jobs have been released before the dark shaded ones, and their deadlines are guaranteed to be prior to $t_{[\star]}$. $\sigma^D_{[\star]}$ is the sum of execution times of the light and dark shaded jobs. Since in overload situations EDF executes jobs non-preemptively, in the order of their working deadline, all light colored jobs in the figure will be executed before $\tau_{j[\star]}$, since their deadlines are before $t_{[\star]}$. The execution times of these jobs represent $\sum_{i \in I} c_{i[\star]} \cdot (q_{i[\star]} - 1)$ out of $\sigma^D_{[\star]}$. From the rest of the jobs considered in $\sigma^D_{[\star]}$ (the dark colored ones), the ones with deadlines smaller than that of $\tau_{j[\star]}$ will be executed before it ($\tau_{1[\star]}$ and $\tau_{2[\star]}$ in the figure), and the rest will be executed after $\tau_{j[\star]}$ ($\tau_{n[\star]}$ in the figure). Also there may exist other, not yet released jobs, that have their deadlines prior to the deadline of $\tau_{j[\star]}$ (e.g. $\tau_{1[\star+1]}$ in Figure 5) which also need to be considered. Taking all of this into account, and considering that $\rho_{i[\star]}$, $\forall i \in I$ are the release rates of all jobs, the response time of $\tau_{j[\star]}$ is:

$$r_{j[\star]} = \sum_{i \in I} c_{i[\star]} \cdot (q_{i[\star]} - 1) + \sum_{i \in I} c_{i[\star]} \cdot \left\lfloor \frac{\frac{1}{\rho_{j[\star]}} + \frac{1}{\rho_{i[\star]}} - \phi_{i[\star]}}{\frac{1}{\rho_{i[\star]}}} \right\rfloor \tag{23}$$

The following theorem gives an upper bound on the response time of the tasks in the system.

Theorem 5: An upper bound on the worst-case response time of the task τ_j in the system can be computed using the following equation:

$$r^{\max}_j = \frac{1}{\rho^{\min}_j} + h \cdot (u^R_{\max} - 1) + \sum_{i \in I} c^{\max}_i \cdot \left\lfloor \frac{\rho^{\max}_i}{\rho^{\min}_j} \right\rfloor \tag{24}$$

◇

Proof: The proof follows from equation (23) by observing that:

1) $\sigma^D_{[\star]} = h \cdot (u^R_{[\star]} - 1) \leq h \cdot (u^R_{\max} - 1)$ (from equation (16a), since the system is overloaded);
2) When $u^R_{\max}$ occurs at time instance $t_{[\star]}$, the last released job of task τ_j is released with at most $1/\rho^{\min}_j$ before $t_{[\star]}$.

■

Up to this point, we have concerned ourselves with the scheduler used, and we have determined a formula for the worst-case response time for EDF assuming knowledge of the largest resource request in the system ($u^R_{\max}$). One should note that the largest resource request in the system typically depends on the scheduler, the controller (f_c) and the prediction of future execution times (f_p) used in the system. The value computed in equations (21) is an upper bound on the largest resource request, since it is independent on these parameters. By adding extra constraints on the scheduler, the controller, or the prediction function, one may be able to tighten this bound. We will now consider two classes of controllers for which we will determine $u^R_{\max}$. The two classes of controllers are:

$$\mathbf{C1} = \left\{ f_c : \mathcal{X} \to \mathbf{P} \middle| \rho_{[k+1]} \in \begin{cases} \{\overline{\rho}^{\min}\}, & \text{if } \Gamma^\alpha_{[k]} = \emptyset \\ \mathbf{P}, & \text{otherwise} \end{cases} \right\} \tag{27}$$

$$\mathbf{C2} = \left\{ f_c : \mathcal{X} \to \mathbf{P} \middle| \rho_{[k+1]} \in \begin{cases} \{\overline{\rho}^{\min}\}, & \text{if } \Gamma^\alpha_{[k]} = \emptyset \\ \Gamma^\alpha_{[k]}, & \text{otherwise} \end{cases} \right\} \tag{28}$$

where $\Gamma^\alpha_{[k]}$ is defined as in equation (25) and $\alpha > 1$ is an arbitrary constant ($\overline{\phi}^p$ is obtained from equation (16b), according with the chosen $\overline{\rho}$). $\Gamma^\alpha_{[k]}$ is the set of all rate vectors which will lead to $u^D_{[k+1]} = \alpha$, if the future average execution times $\overline{c}_{[k+1]}$ are equal to the predicted ones $\overline{c}^p_{[k]}$. To observe this, substitute $u^R_{[k]}$ from equation (14) into equation (15) and rewrite it for $u^D_{[k+1]}$ instead of $u^D_{[k]}$.

The two classes of controllers have the following meaning: **C2** always tries to take decisions such that u^D is kept very aggressively around α. When $u^D_{[k]} \neq \alpha$, the resource demand will be brought back to α as soon as possible ($u^D_{[k+1]} = \alpha$ if the prediction is correct). **C1** is a class of more general controllers, with the same goal. We first show that this classes of controllers lead to stable systems.

Lemma 2: Any system for which $f_c \in \mathbf{C1}$ is stable and $\mathbf{C2} \subset \mathbf{C1}$. ◇

Proof: Since $\Gamma^\alpha_{[k]} \subset \mathbf{P}$ and $\{\overline{\rho}^{\min}\} \subset \Gamma_\star$, $\mathbf{C2} \subset \mathbf{C1}$ follows directly. Also condition (20) is satisfied.

We will now show that for any system having $f_c \in \mathbf{C1}$, f_c also satisfies condition (19) for a certain u^D_Ω. We want to show that there exists a finite u^D_Ω such that, whenever $u^D_{[k]} \geq u^D_\Omega$, it also happens that $\Gamma^\alpha_{[k]} = \emptyset$. We can then say that u^D_Ω is larger than the largest value of $u^D_{[k]}$ for which $\Gamma^\alpha_{[k]} \neq \emptyset$. From equation (16a) we have that $\frac{1}{h} \sum_{i \in I} c_{i[k]} \cdot q_{i[k]} = u^R_{[k]} - 1$ when the system is overloaded.

$$\Gamma^{\alpha}_{[k]} = \Big\{\overline{\rho} \in \mathbf{P}\Big| \sum_{i\in I} c^{p}_{i[k]} \cdot \frac{q^{\mathrm{p}}_{i[k]} + \lceil \rho_i \cdot \max\{0, h - \phi_{i[k]}\}\rceil - \rho_i \cdot \phi^{p}_{i[k]}}{h} = \alpha,\ \overline{c}^{p}_{[k]} = f_p(\overline{c}_{[k]})\Big\} \tag{25}$$

$$u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[k]} + 1 = \sum_{i\in I} c^{\max}_i\big(\frac{q_{i[k-1]}+1}{h} + \rho_{i[k]}\big) = \sum_{i\in I} \frac{c^{\max}_i}{c^{p}_{i[k-1]}} \cdot u_{i[k]} \le \alpha \cdot \max_{i\in I}\Big\{\frac{c^{\max}_i}{c^{\min}_i}\Big\} \tag{26}$$

From this and (25) we have that the largest value of $u^{D}_{[k]}$ happens when $\overline{c}_{[k]} = \overline{c}^{\max}$ and $f_p(\overline{c}_{[k]}) = \overline{c}^{\min}$ and is:

$$u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[k]} + 1 = \frac{1}{h}\sum_{i\in I} c^{\max}_i \cdot q_*$$

where $\overline{q}_*$ is obtained by solving the following linear programming problem:

$$\begin{aligned} &\text{maximize } \sum_{i\in I} c^{\max}_i \cdot q_i \qquad \text{subject to} \\ &q_i \ge 0, \\ &\frac{1}{h}\cdot\sum_{i\in I} c^{\min}_i \cdot q_i + \sum_{i\in I} c^{\min}_i \cdot \rho^{\min}_i = \alpha \end{aligned} \tag{29}$$

where that last constraint is that $\Gamma^{\alpha}_{[k]} \neq \emptyset$. Since u^{D}_{Ω} exists, the proof follows. ∎

Lemma 3: For any system $\{\mathcal{T}, \mathcal{X}, \mathcal{A}, \mathcal{S}\}$ with $f_c \in$ **C1** and $\mathcal{A}$ defined as in equation (22), the largest possible value of the resource request given by

$$u^{R}_{\max} = \sum_{i\in I} c^{\max}_i\big(\frac{q_{*i}+1}{h} + \rho^{\max}_i\big) \tag{30}$$

where $\overline{q}_*$ is the solution of the linear programming problem (29). ◇

Proof: For any state for which $\Gamma^{\alpha}_{[k]} \neq \emptyset$, $u^{D}_{[k]} \le u^{R}_{[k]} \le u^{R}_{\max}$ since $\overline{c}_{[k]} \le \overline{c}^{\max}$ and $\overline{\rho}_{[k]} \le \overline{\rho}^{\max}$, $\forall i \in I$.

For any state for which $\Gamma^{\alpha}_{[k]} = \emptyset$ and $\Gamma^{\alpha}_{[k-1]} \neq \emptyset$, we have that the system is overloading ($u^{D}_{[k]} \ge 1$) and therefore $u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[k]} + 1 \le V_{[k-1]} + 1 \le u^{R}_{\max}$.

For any state for which $\Gamma^{\alpha}_{[k]} = \emptyset$ and $\Gamma^{\alpha}_{[k-1]} = \emptyset$, there exists a previous time moment $t_{[k-p]} \le t_{[k-1]}$ such that either $\Gamma^{\alpha}_{[k-p]} \neq \emptyset$ when $u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[k-p]} + 1$; or $t_{[k-p]} = 1$ and $\Gamma^{\alpha}_{[k-p]} = \emptyset$ when $u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[1]} + 1 = \sum_{i\in I} c^{\max}_i \cdot \rho^{\max}_i$ (since $q^{\mathrm{p}}_{[1]} = q_{[0]} = 0$). In both cases $u^{D}_{[k]} \le u^{R}_{[k]} \le u^{R}_{\max}$. From the above cases, since k is arbitrary, the proof follows. ∎

Lemma 4: For any $\{\mathcal{T}, \mathcal{X}, \mathcal{A}, \mathcal{S}\}$ with $f_c \in$ **C2** and $\mathcal{A}$ defined as in equation (22), the largest possible value of the resource request is given by

$$u^{R}_{\max} = \max\Big\{\alpha \cdot \max_{i\in I}\Big\{\frac{c^{\max}_i}{c^{\min}_i}\Big\},\ \sum_{i\in I} c^{\max}_i \cdot \rho^{\max}_i\Big\} \tag{31}$$
◇

Proof: We have two cases to analyze. When $\Gamma^{\alpha}_{[k-1]} \neq \emptyset$ then we have from equations (25) and (28):

$$\sum_{i\in I} c^{p}_{i[k-1]}\big(\frac{q_{i[k-1]}}{h} + \rho_{i[k]}\big) = \alpha$$

and there exist the quantities $u_{i[k]} \in [0, \alpha]$ with $\sum_{i\in I} u_{i[k]} = \alpha$ such that:

$$u_{i[k]} = c^{p}_{i[k-1]}\big(\frac{q_{i[k-1]}}{h} + \rho_{i[k]}\big)$$

From the above, the inequality (26) follows.

On the other hand, when $\Gamma^{\alpha}_{[k-1]} = \emptyset$, there must exist a previous time instance $t_{[k-p]} \le t_{[k-1]}$ with $\Gamma^{\alpha}_{[k-p+r]} = \emptyset$, $\forall r = \overline{0, p-1}$. In this case there are two possibilities: either $t_{[k-p]} = 1$ when $u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[1]} + 1 = \sum_{i\in I} c^{\max}_i \cdot \rho^{\max}_i$; or there exists $\Gamma^{\alpha}_{[k-p-1]} \neq \emptyset$. In this second case we have that $u^{D}_{[k]} \le u^{R}_{[k]} \le V_{[k-p]} + 1$ and inequality (26) applies. ∎

The bound on the worst case response time (for an EDF scheduler) can be calculated using the equation (24) where $u^{R}_{\max}$ is computed according with equation (30) for controllers in **C1**, and equation (31) for controllers in **C2**. These results were determined considering that the system is modeled using the general model. For the constrained model, the classes of controllers **C1** and **C2** can also be defined as in equations (27) and (28), where $\Gamma^{\alpha}_{[k]}$ is:

$$\Gamma^{\alpha}_{[k]} = \Big\{\overline{\rho} \in \mathbf{P}\Big| \sum_{i\in I} c^{p}_{i[k]}\big(\frac{q_{i[k]}}{h} + \rho_i\big) = \alpha;\ \overline{c}^{p}_{[k]} = f_p(\overline{c}_{[k]})\Big\} \tag{32}$$

By a similar reasoning, these classes of controllers can also be shown to be stable, and their worst case resource requests ($u^{R}_{\max} = u^{D}_{\max}$ for the constrained model) are the same as for **C1** and **C2** for the general model (equation (30) for **C1** and equation (31) for **C2**).

VIII. Discussion

In the previous sections we have developed models and stability conditions, first for a constrained system and then for a general one. Obviously, eliminating the constraint on the controller period is very important since it allows to adapt controller rates to the particularities of the application and, thus, to improve the quality of management. In order to provide further insight, we will now compare the controllers generated with the two models in terms of the bounds on the worst case response time of the tasks. We will consider the classes of controllers **C1** and **C2** for both models.

We will construct two test-cases, both consisting of two tasks. The first is characterized by small variations of execution times ($c^{\max}_i/c^{\min}_i = 2$) and rates ($\rho^{\max}_i/\rho^{\min}_i = 2$) and the fact that the rates of both tasks have the same order of magnitude. The second test-case will have large variations ($c^{\max}_i/c^{\min}_i = 10$, $\rho^{\max}_i/\rho^{\min}_i = 10$) and the tasks will have rates of different orders of magnitude.

Table I
RESPONSE TIMES FOR OUR EXAMPLES, CONSIDERING THREE DIFFERENT CONTROLLER PERIODS

Model	Controller	Example 1			Example 2		
		h	$r_1^{\max}$	$r_2^{\max}$	h	$r_1^{\max}$	$r_2^{\max}$
Constrained	**C1**	20	48	53	5000	91200	97050
	C2	20	28	33	5000	46200	52050
		h	$r_1^{\max}$	$r_2^{\max}$	h	$r_1^{\max}$	$r_2^{\max}$
General	**C1**	4	19	24	1000	19750	25600
	C2	4	12	17	1000	10200	16050
	C1	2	15	20	100	3550	9400
	C2	2	10	15	100	2100	7950

Example 1: We consider a task set $\Lambda_1 = \{\tau_1, \tau_2\}$, where:

$$\tau_1 = \{\mathbf{P}_1 = [0.5, 1], \mathbf{C}_1 = [0.5, 1]\}$$
$$\tau_2 = \{\mathbf{P}_2 = [0.25, 0.5], \mathbf{C}_2 = [1, 2]\}$$ ▫

Example 2: We consider a task set $\Lambda_2 = \{\tau_1, \tau_2\}$ where:

$$\tau_1 = \{\mathbf{P}_1 = [0.01, 0.1], \mathbf{C}_1 = [5, 50]\}$$
$$\tau_2 = \{\mathbf{P}_2 = [0.001, 0.01], \mathbf{C}_2 = [50, 500]\}$$ ▫

For the constrained model, as explained in Section V-A, the controller period must be much larger than the largest task period in the system. Let us consider that this assumption holds if h is no smaller than 5 times the largest task period. For the general model this constraint disappears, and we can choose smaller controller periods as well. In Table I we present the response time bounds $r_i^{\max}$ for our chosen examples, considering three controller periods h:

$$5 \cdot \max_{i \in I} \left\{ \frac{1}{\rho_i^{\min}} \right\}; \quad \max_{i \in I} \left\{ \frac{1}{\rho_i^{\min}} \right\}; \quad \text{and} \quad \min_{i \in I} \left\{ \frac{1}{\rho_i^{\min}} \right\}$$

the first period is used for the constrained model, and the following two for the general one.

We have two observations to make:

1) in both examples the class of more aggressive controllers (**C2**) is able the control the system such that response times reduce up to half when compared with **C1**; this is due to the fact that **C2** is more aggressive in controlling the resource demand of the system.
2) in Example 1, the best performing controller (in terms of response time) **C2** with $h = 2$ leads to worst case response times that are 21% (for τ_1) and 28% (for τ_2) of the worst case response times of the worst performing controller (**C1** with $h = 20$). In the second example, the performance gap is much more dramatic, and **C2** with $h = 100$ leads to worst case response times that are 2.3% and 8.2% of the worst case response times of **C1** with $h = 5000$. This is largely due to the fact that for the worst performing controllers (modeled according with the constrained model) the period of the controller h must be much larger than the largest period in the system (our assumption from Section V-A), and in overloaded conditions, a task with high rate will release a high number of jobs, during the interval h, that will only queue up.

The general model removes this constraint and the controllers for this model, can take advantage of much smaller controller periods, in order to reduce response times by very large amounts.

IX. CASE STUDY OF THREE RESOURCE MANAGERS

In this section, we take three resource management policies, presented in previous literature, and determine if they lead to stable real-time systems. We will consider the *QRAM* algorithm described in [1], and the *corner-case* and *QoS derivative* algorithms described in [16].

The *QRAM* and *corner-case* algorithms work in similar ways. They consider as a resource the processor utilization and, when the control algorithms are executed, they initially select task rates such that the perceived resource demand (computed based on estimations of future job execution times, and task queues) is minimum ($\overline{\rho}^{\min}$). Then, if resources are still available in the system, the algorithms select tasks whose rates are increased until all available resources are used. This tasks are selected such that the value of some quality-of-service function is maximized. If no resources are available, the new task rates are $\overline{\rho}^{\min}$. From this behavior we can observe that *QRAM* and *corner-case* satisfy Theorem 4 and, therefore, these resource managers lead to stable systems. More specifically, these resource managers belong to the class **C2**.

The *QoS derivative* algorithm works in a different way. At the beginning it determines the resource demand in the system, assuming current rates. Then the manager solves a convex optimization problem, whose goal is to select new task rates such that some quality-of-service function is maximized, with the constraint that the resource demand with the new rates is equal with the amount of available resources. If a feasible solution (a solution that satisfies the constraint) exists, the system is stable. However, if the constraint cannot be satisfied by any $\overline{\rho} \in \mathbf{P}$, it cannot be demonstrated that the selected rates will satisfy Theorem 4. A straight forward approach would be to test the solution of the optimization to determine if it is feasible, and if not, to select $\overline{\rho} \in \Gamma_\star$. However, for the convex optimization to find a feasible solution, it is required that the starting point satisfies the constraint[4] otherwise no feasible solution will be found and, in practice, the straight forward approach always sets rates in $\Gamma_\star$, which will keep the processor load unnecessarily low.

Our solution is to modify the *QoS derivative* algorithm to start from a feasible point (if one exists):

Input: $c_{i[k]}$, $q_{i[k]}$, $\rho_{i[k]}$, h
1: /* here we assume that $u_{[k]}^D$ is computed according to Equation (15) */
2: $u_i^D \leftarrow \frac{1}{h} \cdot c_{i[k]} \cdot q_{i[k]} + c_{i[k]} \cdot \rho_{i[k]}$

[4]In addition to this the Karush-Kuhn-Tucker matrix must be non-singular at the starting point, but it can be shown that this condition holds for any $\overline{\rho} \in \mathbf{P}$. See [19] Sec. 10.2 for an in depth treatment of these conditions.

3: $\Delta u \leftarrow u - u^D_{[k]}$
4: **while** $i < n$ and $\Delta u \neq 0$ **do**
5: /* change $\rho_{i[k]}$ to ρ_{i*} such that the absolute value of Δu is reduced */
6: $\Delta u \leftarrow \Delta u - c_{i[k]} \cdot \rho_{i[k]} + c_{i[k]} \cdot \rho_{i*}$
7: $i \leftarrow i + 1$
8: **end while**
9: $\overline{\rho}_{[k+1]} \leftarrow$ *QoS derivative*$(\overline{\rho}_*)$
10: **return** $\overline{\rho}_{[k+1]}$

The algorithm changes the initial rates to a new vector $\overline{\rho}_*$ (line $4-8$). The original manager is then called (line 9 in the algorithm) with this rate vector, as a starting point If there are $\overline{\rho} \in \mathbf{P}$ for which the constraint is satisfied, then the algorithm has a feasible solution, otherwise, the algorithm will set $\overline{\rho}_{[k+1]} = \overline{\rho}^{\max}$ when the system is underloading and $\overline{\rho}_{[k+1]} = \overline{\rho}^{\min}$ when the system is overloading. From this behavior we can observe that the *QoS derivative* resource manager, with the modification, is also stable and belonging to **C2**. Also note that if quality-of-service is not an issue, the above algorithm, with line 9 changed to $\overline{\rho}_{[k+1]} \leftarrow \overline{\rho}_*$ is also guaranteed to be stable and belonging to **C2**.

X. Conclusions

In many real-time systems with large variations in execution times, it is important to regulate and manage the utilization of system resources at runtime. An important issue at design time is to verify that the real-time system is stable when using a certain resource manager. Stability means that the resource demand is bounded under all runtime scenarios. We have developed a model for real-time systems and used it to derive comprehensive conditions that resource managers must comply with, in order to render the system stable. For the derived models we also derived bounds on the response times of tasks. We have applied our results to existing resource managers to verify their correctness in terms of stability.

References

[1] C. Lee, J. Lehoczky, R. Rajkumar, D. Siewiorek. *"On Quality of Service Optimization with Discrete QoS Options."* In proceedings of Real-Time Technology and Applications Symposium, pp.276, 1999.

[2] G. C. Buttazo, G. Lipari, L. Albeni. *"Elastic Task Model for Adaptive Rate Control."* In Proceedings of the IEEE Real-Time Systems Symposium, pp. 286, December 1998.

[3] G. C. Buttazo, L. Albeni. *"Adaptive Workload Management through Elastic Scheduling."* Journal of Real-Time Systems, vol. 23, pp. 7-24, July 2002.

[4] G. C. Buttazo, M. Velasco, P. Marti and G. Fohler. *"Managing Quality-of-Control Performance Under Overload Conditions."* In Proceedings of the Euromicro Conference on Real-Time Systems, pp. 53-60, July, 2004.

[5] M. Marioni, G. C. Buttazo. *" Elastic DVS Management in Processors With Discrete Voltage/Frequency Modes."* IEEE Transactions on Industrial Informatics, vol. 3, pp. 51-62, February, 2007.

[6] C. Lu, J. A. Stankovic, S. H. Son, G. Tao. *"Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms."* Real-Time Systems, vol. 23, pp. 85-126, 2002.

[7] L. Palopoli, T. Cucinotta, L. Marzario, G. Lipari. *"AQuoSA – adaptive quality of service architecture."* Journal of Software–Practice and Experience, vol. 39, pp. 1-31, 2009.

[8] T. Cucinotta, L. Palopoli. *"QoS Control for Pipelines of Tasks Using Multiple Resources."* IEEE Transactions on Computers, vol. 59, pp. 416-430, 2010.

[9] J. Combaz, J. C. Fernandez, J. Sifakis, L. Strus. *"Symbolic Quality Control for Multimedia Applications."* Real-Time Systems, vol. 40, pp. 1-43, October, 2008.

[10] X. Liu, X. Zhu, P. Padala, Z. Wang, S. Singhal. *"Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform."* In Proceedings of the Conference on Decision and Control, pp. 3792-3799, December 2007.

[11] J. Yao, X. Liu, M. Yuan, Z. Gu. *"Online Adaptive Utilization Control for Real-Time Embedded Multiprocessor Systems."* In Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, pp. 85-90, 2008.

[12] T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, Y. Lu. *"Feedback performance Control in Software Services – Using a Control-Theoretic Approach to Achieve Quality of Service Guarantees."* IEEE Control Systems Magazine, vol. 23, pp. 74-90, June 2003.

[13] A. Cervin, J. Eker, B. Bernhardsson, K. E. Årzén. *"Feedback-Feedforward Scheduling of Control Tasks."* Real-Time Systems, vol. 23, pp. 25-53, July, 2002.

[14] A. Cervin, J. Eker. *"The Control Server: A Computational Model for Real-Time Control Tasks."* Proceedings of the 15th Euromicro Conference on Real-Time Systems, July 2003.

[15] C. L. Liu, J. W. Layland. *"Scheduling algorithms for multiprogramming in hard-real-time environment."* Journal of ACM, pp. 40-61, 1973.

[16] S. Rafiliu, P. Eles, and Z. Peng. *"Low Overhead Dynamic QoS Optimization Under Variable Execution Times."* Proceedings of 16th IEEE Embedded and Real-Time computing Systems and Applications (RTCSA), pp. 293-303, 2010.

[17] A. N. Michel, L. Hou, D. Liu. *"Stability of Dynamical Systems: Continuous, Discontinuous, and Discrete Systems."* Birkhäuser Boston, 2008.

[18] E. Kreyszing. *"Introduction to Functional Analysis with Applications."* John Wiley & Sons., Inc, 1989.

[19] S. Boyd, L. Vandenberghe. *"Convex Optimization."* Cambridge University Press, 2008.