An Efficient Periodic Resource Supply Model for Workloads with Transient Overloads

Akramul Azim
Department of Electrical and Computer Engineering
University of Waterloo, Canada
aazim@uwaterloo.ca

Shreyas Sundaram
Department of Electrical and Computer Engineering
University of Waterloo, Canada
ssundara@uwaterloo.ca

Sebastian Fischmeister
Department of Electrical and Computer Engineering
University of Waterloo, Canada
sfischme@uwaterloo.ca

***Abstract*—Real-time applications have deadline constraints. The system should provision sufficient resources for the application to meet the deadlines, and use supply and demand bound functions to analyze the schedulability of workloads. The concept of the demand bound function describes the upper bound on the resources required by the application, while the supply-bound function specifies the lower bound on the resources supplied to the tasks. If the system provides fewer resources than required, the application will experience an overload. Most work concentrates on designing systems that cannot experience short periods of overloads.**

This work explores resource provisioning for control applications that can tolerate overloads. It introduces analysis techniques for supply and demand bound functions that specifically consider overloads and delays in a periodic resource model. With this extended model, the work addresses three problems: (1) determine the worst-case delay for a given resource demand and supply under a periodic resource model, (2) find a periodic resource supply for a given workload and worst-case tolerable delay, and (3) for a control system with a given robustness criterion, identify a periodic resource supply with a worst-case delay.

I. Introduction

Real-time applications require sufficient resources (i.e., computation, communication) to meet their timing requirements. Given a set of applications, various schedulability tests exist to evaluate whether each application receives the required resources. For instance, in hard real-time systems, a schedulability test, based on supply and demand bound functions [26], [1], [21], [18], will indicate that insufficient resources are available when the supply-bound function (*sbf*) drops below the demand bound function (*dbf*) for a given time interval. However, this condition is too conservative for systems such as soft real-time systems or firm real-time systems that can tolerate the occurrence of overloads (i.e., situations where the application requires more resources than available) with a bounded duration. For example, if the system can tolerate dropping or delaying tasks, the schedulability test will accept a system whose supply bound function intermittently drops below the demand bound function.

Overloads can be either transient or permanent. A transient overload always has a bounded time span until its resolution and can occur perhaps due to excessive task execution times, or due to the simultaneous arrival of asynchronous events. A permanent overload always has an unbounded time span and will occur if the system is badly designed and unschedulable. A large class of systems can tolerate the occurrence of overloads with a specified duration. For example, while control systems require a certain measure of reliability in their feedback loops to function correctly, these systems are usually robust to short delays in their control updates (depending on the dynamics of the system) [31], [13]. Other applications that fit into this category are classic soft real-time applications such as video-on-demand that can show a static image for a few frames when overloaded, or audio applications.

Compositionality is a way to convert multiple independent timing requirements of different components of a system into a single real-time system timing requirement. Satisfying timing requirements together with logical correctness can make a real-time system predictable, guaranteeing that it will operate correctly when deployed. Abstracting timing at the higher level of system design permits designers to model the system by eliminating the low-level timing requirements of different components in cyber-physical systems. Since it is often preferable to work at higher levels of abstraction, compositionality analysis facilitates design productivity.

Motivated by feedback control systems that are robust to small delays in the feedback loop, this work investigates an efficient periodic resource supply model for workloads that can tolerate transient overloads and delays and extends *dbf* and *sbf*-based techniques to analyze such systems. To characterize overloads, this paper focuses on the worst-case delay for recovery. Thereafter, we provide methods for finding a suitable resource to meet desired delay constraints.

The novelty of the work lies in finding an efficient resource model for tasks with transient overloads. This work extends the analysis of periodic resource models using the supply and demand bound functions as described in [25] by introducing overloads. Consequently, this extended model can find the resource supply required for a hierarchical or compositional system with transient overloads. In this work, we mainly focus on the application to control systems. The workflow of the application to control systems is the following: (1) a control engineer models a physical system and defines the control objective, (2) the control engineer designs a feedback controller for the plant based on a given sampling period, (3) the control engineer determines the maximum tolerable delay in calculating and applying the feedback inputs, based on the dynamics of the system, and (4) the system provisions the computational resources required to perform the computations

within the given delay. This work uses the earliest deadline first (EDF) scheduling scheme, leaving the extension to other schemes for our subsequent work.

The remainder of this paper is structured as follows: Section II presents an overview of the problem, and the motivating application of control systems is presented in Section III. This produces a set of computational and tuning requirements, along with a specification on the maximum overload (i.e., delay) that can be tolerated. Section IV presents the system model and shows how to characterize overloads using supply and demand bound functions. Section V presents methods on calculating a suitable resource supply to meet the workload timing requirements in the presence of overloads. Section VI demonstrates the use of the workflow for state feedback control of two plants. Section VII presents some related work on schedulability analysis, specifically using supply and demand bound functions. Section VIII discusses what parameters in the system model affect the overloads, and Section IX concludes the paper.

II. Problem Statement

A supply bound function (*sbf*) and demand bound function (*dbf*) convert the timing requirements of the workload and the resource supply into a single timing requirement. Traditionally and informally, the *dbf* must stay below the *sbf* in all time intervals in order to avoid overloads. When permitting overloads, the *sbf* can remain below the *dbf* for bounded time intervals. Overloads then cause delays as the application must wait to receive sufficient resources. The worst-case delay δ^* is the maximum delay that tasks experience before they receive their requested resources.

In the context of control systems, a scheduling framework that supports overloads can help control engineers design efficient and safe systems. A control system task-specification might include the worst-case tolerable delay in all time intervals. One can design an efficient resource supply to exploit the robustness of a given set of tasks to delays. Then, the following problem identifies the resource supply that the system designer needs to provide for the control application that permits overloads:

Goal: Given a control system workload $W = \{T_1, T_2, \ldots, T_n\}$ and a worst-case delay (δ^*), find the resource supply such that W is schedulable under the *EDF* scheduling policy and experiences a worst-case delay of at most δ^*.

A solution to this problem is applicable to hard, soft, and firm real-time systems. For a hard real-time system, δ^* must be zero. Soft real-time systems might specify some bound for δ^*. Firm real-time systems [20] may specify a δ^* with a probability of occurring.

III. Feedback Control Systems With Delays

Consider a physical system (plant) modeled as a linear time-invariant system of the form

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{1}$$

where t is the time variable, $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in \mathbb{R}^m$ is the control input applied via the actuators, and matrices A and B are of appropriate dimensions. This model is obtained from the physical processes governing the system under consideration. A typical objective is to choose the input $u(t)$ so that the system is *stable* in the following sense.

Definition 1: The system (1) is said to be asymptotically stable if $\lim_{t\to\infty} x(t) = 0$ for any initial condition $x(0)$.

When the full state $x(t)$ is measurable and the pair (A, B) satisfies an algebraic property known as *stabilizability* [6], it is possible to find a state-feedback control input of the form $u(t) = Kx(t)$ such that the system is asymptotically stable (where K is an appropriate $m \times n$ gain matrix). When the plant is controlled over a network, however, stabilization is complicated by issues such as sampling, delays and packet drops. There has been a large amount of research devoted to characterizing conditions under which stabilization is possible, for various assumptions on the system and the network [31], [13], [7], [14], [12], [23], [22]. This work follows the approach in [7], which presented a general and computationally efficient method to obtain bounds on the delays that can be tolerated by a given control system.

First, we assume that the plant state is sampled every p seconds to produce the state measurements $x(t_k)$, where $t_k = kp$ for $k \in \mathbb{N}$. These state measurements are then sent over the network to the controller (i.e., a computational resource), which calculates the control input $u(t_k) = Kx(t_k)$ and sends this value to the plant's actuators, where it is held until the next input is received. There is a delay τ_k incurred between measuring the plant's state at time t_k and applying the input $u(t_k)$. Thus, as in [7], the system evolves as follows:

$$\dot{x}(t) = Ax(t) + BKx(t_k), \;\; t \in [t_k + \tau_k, t_{k+1} + \tau_{k+1}).$$

Let τ^* be the maximum possible delay over the network (i.e., $\tau^* = \sup_{k\in\mathbb{N}} \tau_k$). The following result from [7] provides a method to determine whether the system will be stable with a given worst-case delay and feedback gain K.

Theorem 1 ([7]): For a given scalar η and matrix K, if there exist matrices $P > 0$, $T > 0$, N_i and M_i $(i = 1, 2, 3)$ of appropriate dimensions such that (3) is true, then the system is asymptotically stable with the state feedback input $u(t) = Kx(t_k)$, $t \in [t_k + \tau_k, t_{k+1} + \tau_{k+1})$, as long as the sampling period p and worst-case delay τ^* satisfy

$$p + \tau^* \leq \eta. \tag{2}$$

For a square symmetric matrix P, the notation "$P > 0$" in the above theorem indicates that the matrix is *positive definite*. The matrix in (3) is symmetric, and to save space, the $*$ symbols are used as placeholders for the appropriate matrix elements. When η is a fixed constant, the above expression is a *Linear Matrix Inequality*, which can be solved efficiently for the unknown matrices P, T, N_i and M_i using convex programming software such as CVX [4], [11]. One can find the largest value of η for which the system will be stable by using bisection.

To relate the above characterization of stability to the characterization of overloads or delays, we note that the worst-case delay δ^* represents the longest length of time *after the end of any task's period* that would be required for the necessary

$$\begin{bmatrix} N_1 + N_1^T - M_1A - A^TM_1^T & N_2^T - N_1 - A^TM_2^T - M_1BK & N_3^T - A^TM_3^T + M_1 + P & \eta N_1 \\ * & -N_2 - N_2^T - M_2BK - K^TB^TM_2^T & -N_3^T + M_2 - K^TB^TM_3^T & \eta N_2 \\ * & * & M_3 + M_3^T + \eta T & \eta N_3 \\ * & * & * & -\eta T \end{bmatrix} < 0. \quad (3)$$

computational resources to become available. Thus, from the perspective of the control system, the longest possible delay seen by a packet generated at time t_k would be $\tau^* = p + \delta^*$ (i.e., the length of one period plus the maximum additional time required to obtain the desired resources). Thus, once we find a worst-case value for η from Theorem 1, we can obtain an upper bound on δ^* from Equation (2) as

$$\delta^* = \eta - 2p.$$

Figure 1 outlines how developers can use the results of this work. After the control engineer designs the system, she computes the *dbf* of the application, and specifies the upper bound on the worst-case delay (for example, using the technique described above). Second, using our algorithms as specified in Section V, the engineer finds a resource supply for the resource of interest (e.g., the processor). Third, the engineer analyzes the *sbf*, the *dbf*, and the control system to determine whether the found supply is sufficient for the system (e.g., worst-case delay remains below the specified bound). If the found supply fits the system, then in the fourth step, the engineer can use the supply to deploy the system; otherwise, the engineer can tweak the constraints on the supply generation and find a different supply.

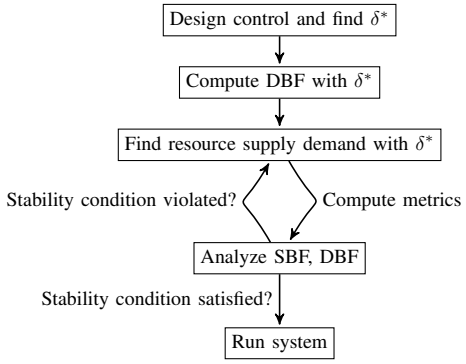

Fig. 1. Workflow for our approach

IV. Overloads in Supply and Demand Curves

The system model consists of a periodic resource model and a periodic workload, consisting of a set of tasks (e.g., the tasks to process the feedback control signals). A task T_i is characterized by a tuple (p_i, e_i) where p_i is the period and e_i is the worst-case execution time. We assume the deadline d_i of task i is equal to p_i. A set of tasks or a workload is characterized by a set of tuples: $\{(p_1, e_1), \ldots, (p_n, e_n)\}$. This work assumes that all tasks in the system are fully preemptive and have a known tolerable delay. The hyperperiod of all tasks' periods forms the cycle at which the system repeats its behaviour. A periodic resource model indicates resource replenishment in each period. Given a periodic resource model $R(\lambda, \theta)$, tasks are allocated for θ time units in every interval $[k\lambda, (k+1)\lambda], k \in \mathbb{N}$. This work uses the *EDF* scheduling policy. The generalization of our work to other scheduling policies is the object of ongoing research.

A. Characterizing Overloads

Since a periodic resource $R(\lambda, \theta)$ guarantees a supply of at least θ time units in every interval $[k\lambda, (k+1)\lambda], k \in \mathbb{N}$, the model leaves open how the guaranteed θ time units are distributed over a time interval of size λ. An instance of the periodic resource is a time trace of resource allocations that satisfies the guarantee (λ, θ). For a given workload, the *dbf* is the maximum possible resource demand in any time interval t. Obtained from [26], Equation 4 shows how to calculate the resource demand for n tasks for the *EDF* scheduling scheme for a time interval of length t:

$$dbf(W, EDF, t) = \sum_{T_i \in W} \left\lfloor \frac{t}{p_i} \right\rfloor e_i. \quad (4)$$

The supply bound function (*sbf*) calculates the minimum resource supply in any time interval t. Using Equation 5 from [26], it is possible to calculate the periodic resource supply in any time interval t as

$$sbf(t) = \begin{cases} (t - (k+1)(\lambda - \theta)) & \text{if } t \in [k_1, k_2] \\ (k-1)\theta & \text{otherwise} \end{cases} \quad (5)$$

with $k_1 = (k+1)\lambda - 2\theta$, $k_2 = (k+1)\lambda - \theta$, and k as

$$k = \max(\lceil (t - (\lambda - \theta))/\lambda \rceil, 1). \quad (6)$$

Note that, the value of k is greater than or equal to 1.

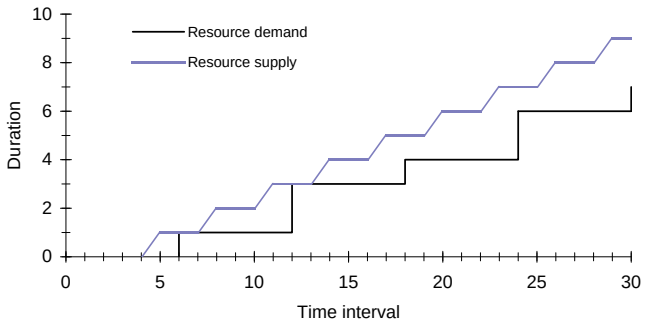

Fig. 2. An example of *sbf* and *dbf* where $\forall t$: *sbf* $(t) \geq$ *dbf* (W, t, EDF)

Definition 2 (Overload): An overload is said to occur in a time interval t when $\exists t > 0 : dbf(t) \geq sbf(t)$.

While previous work on the periodic resource model [28] only discusses resource supplies and demands for which the

sbf is always less than the *dbf*, our work focuses on using the periodic resource model in systems for which the *dbf* can be greater than the *sbf* for some time intervals. Figure 3 shows such overloads. Figure 4 shows a more detailed view of a single overload.

Example 1: Consider a periodic resource supply $R(3,1)$, and a scheduling model $M(W,R,EDF)$ that has two tasks in the workload, $W = \{T_1(6,1), T_2(12,1)\}$. Figure 2 shows the computed *sbf* and *dbf*. This workload is schedulable with the given resource, because the supply is always greater than the demand.

Slightly changing the workload to $W = \{T_1(6,1), T_2(12,2)\}$ makes the system infeasible to schedule. Figure 3 shows the new *dbf*. The system is infeasible because in a time interval of $t = 12$, the system can experience an overload, since the *dbf* is greater than the *sbf*.

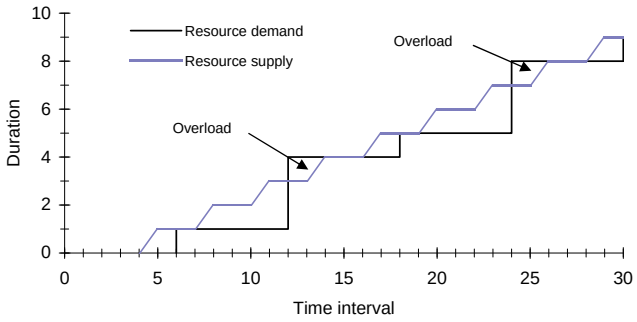

Fig. 3. An example of an overload ($\exists t$: *sbf* (t) < *dbf* (W,t,EDF))

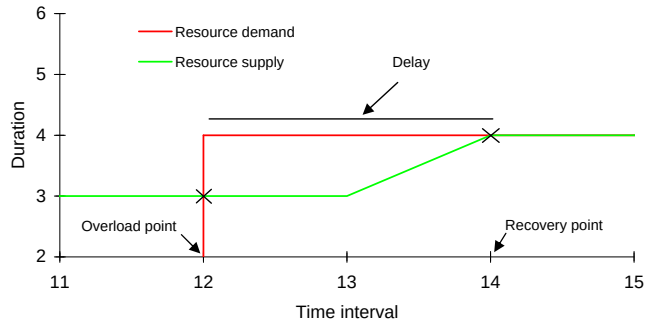

Fig. 4. A detailed view of an overload shown in Figure 3

An overload starts and ends at a specific time interval at which the *sbf* and the *dbf* intersect before the *dbf* becomes greater than the *sbf*. The points at which this intersection happen are called *points of interest*.

Definition 3 (Points of Interest): A point t will be an *overload point* (t_o), if

$$\exists \pi > 0 : \forall \epsilon \in (0,\pi]\ \textit{sbf}(t-\epsilon) \geq \textit{dbf}(t-\epsilon) \text{ and } \textit{sbf}(t) < \textit{dbf}(t). \quad (7)$$

A point t will be a *recovery point* (t_r), if

$$\exists \pi > 0 : \forall \epsilon \in (0,\pi]\ \textit{sbf}(t-\epsilon) < \textit{dbf}(t-\epsilon) \text{ and } \textit{sbf}(t) = \textit{dbf}(t). \quad (8)$$

We use these points of interest to define the *duration* and *severity* of an overload. The first recovery point with a time interval greater than an overload point is the *associated* recovery point. Informally, this is the point at which the *sbf* catches up to the *dbf* again.

Definition 4 (Overload duration and severity): For a given overload point t_o and its associated recovery point t_r such that $t_o \leq t_r$, the *duration* of an overload is $t_r - t_o$ when $t_o \geq t$ or $t - t_o$ when $t_r < t_o \leq t$, where t is the length of the time interval under consideration. The *severity* of the overload is $\textit{dbf}(t_o) - \textit{sbf}(t_o)$.

The duration of an overload is a useful metric when designing the system. The existence of a given overload point t_o and its associated recovery point t_r means there exist time intervals of length t_r in which the system may be overloaded. However, at the same time, for time intervals greater or equal to t_r, the system no longer experiences an overload. Thus the difference $t_r - t_o$ specifies the *delay* that tasks experience when waiting for their resources. However, if no such recovery point t_r exists, the delay is taken to be $t - t_o$, where t is the time interval under consideration.

Observation [Duration=Delay]: For a given overload point t_o and its associated recovery point t_r, the duration of the overload $t_r - t_o$ or $t - t_o$ characterizes the delay that tasks experience during the overload before receiving their demanded resource.

Definition 5 (Worst-case Delay): For a scheduling model $M(W,R,EDF)$ with a periodic resource $R(\lambda,\theta)$ and a workload W, then under the *EDF* policy, the maximum of all overload durations will be the worst-case delay (δ^*) of any task.

The following observations limit the locations of points of interest:

1) For an overload point t_o, the *dbf* can only exceed the *sbf* at points t where the *dbf* increases. The *dbf* only increases at $t = m \cdot p_i$ for some positive integer m and task period p_i. Thus, if the system contains overload points, then they must be values in the set $\{mp_i : m \in \mathbb{N}^+\}$.
2) For a recovery point t_r, the *sbf* can only be equal to or greater than the *dbf* at points t where the *sbf* is increasing, i.e., at $t = c\lambda + 2(\lambda - \theta) + r$, where c is a positive integer and $1 \leq r \leq \theta$. Thus, if the system contains recovery points, then they must be of this form.

We now show that when the resource utilization is equal to the workload utilization, the overload characteristics of the system are periodic. In the process, we characterize the length of the largest time interval that has to be considered to analyze the system. To do this, we define the function

$$f(t) = \textit{sbf}(t) - \textit{dbf}(t), \quad t \in \mathbb{R}_{\geq 0}. \quad (9)$$

Note that the values of t for which $f(t) < 0$ correspond exactly to time intervals where the system is experiencing an overload.

Theorem 2: Consider a system with workload utilization $U_W = \sum_i \frac{e_i}{p_i}$ and resource utilization $U_R = \frac{\theta}{\lambda}$. If $U_R = U_W$, then after $t = 2(\lambda - \theta)$, the function $f(t)$ is periodic with period $\text{LCM}(\lambda, p_1, \ldots, p_n)$, i.e.,

$$f(2(\lambda-\theta) + t + k\text{LCM}(\lambda, p_1, \ldots, p_n)) = f(2(\lambda-\theta)+t), \quad \forall t \in \mathbb{R}_{\geq 0}, \forall k \in \mathbb{N}.$$

Proof: First, one can verify from Equations (4) and (5) that *dbf*(t) and *sbf*(t) satisfy

$$\textit{dbf}(t + k\text{LCM}(p_1, \ldots, p_n)) = \textit{dbf}(t) + k\text{LCM}(p_1, \ldots, p_n)U_W, \forall t \in \mathbb{R}_{\geq 0}, \forall k \in \mathbb{N}$$

$$\textit{sbf}(2(\lambda-\theta) + t + k\lambda) = \textit{sbf}(2(\lambda-\theta)+t) + k\theta, \quad \forall t \in \mathbb{R}_{\geq 0}, \forall k \in \mathbb{N}.$$

Using these identities, we obtain

$$\begin{aligned} &f(2(\lambda-\theta) + t + k\text{LCM}(\lambda, p_1, \ldots, p_n)) \\ &= \textit{sbf}(2(\lambda-\theta) + t + k\text{LCM}(\lambda, p_1, \ldots, p_n)) \\ &\quad - \textit{dbf}(2(\lambda-\theta) + t + k\text{LCM}(\lambda, p_1, \ldots, p_n)) \\ &= \textit{sbf}(2(\lambda-\theta)+t) + k\text{LCM}(\lambda, p_1, \ldots, p_n)\frac{\theta}{\lambda} \\ &\quad - \textit{dbf}(2(\lambda-\theta)+t) - k\text{LCM}(\lambda, p_1, \ldots, p_n)U_W. \end{aligned}$$

When $U_R = U_W$, this expression becomes

$$\begin{aligned} &f(2(\lambda-\theta) + t + k\text{LCM}(\lambda, p_1, \ldots, p_n)) \\ &= \textit{sbf}(2(\lambda-\theta)+t) - \textit{dbf}(2(\lambda-\theta)+t) \\ &= f(2(\lambda-\theta)+t), \end{aligned}$$

which proves the theorem. ∎

The function $f(t)$ fully captures the relative behavior of the supply bound function and the demand bound function, and the entire function $f(t)$ is characterized by its values in the interval $[0, 2(\lambda-\theta)+\text{LCM}(\lambda, p_1, \ldots, p_n))$. Thus, we only need to analyze the system for intervals up to this length to determine schedulability. We will use this fact in the rest of the paper.

B. Computing the Points of Interest

Overloads can only occur at the points where the *dbf* increases, because the *sbf* is a monotonically increasing function. A recovery can only occur at points where the *sbf* increases and the *dbf* remains flat. Since overload points and recovery points are located at intersection points, it is possible to find these points by solving the equation *sbf*$(t) =$ *dbf*(W, EDF, t):

$$\begin{cases} \sum_{T_i \in W} \lfloor \frac{t}{p_i} \rfloor e_i = (t - (k+1)(\lambda-\theta)) & \text{if } t \in [k_1, k_2] \\ \sum_{T_i \in W} \lfloor \frac{t}{p_i} \rfloor e_i = (k-1)\theta & \text{otherwise.} \end{cases} \tag{10}$$

We use the algorithms (Algorithm 1 and Algorithm 2) to find the overload and recovery points to Equation 10. Algorithm 1 identifies all overload points in any interval of length t. Algorithm 2 computes the recovery points associated with each overload point. The algorithm uses Equation 8 to determine the points.

Algorithm 1 Finding all overload points in intervals of length up to t

Output: Ordered list of overload points L_o

```
for every i → 1, ..., |W| do
    for every 1 ≤ m ≤ ⌊t/p_i⌋ do
        if mp_i satisfies (7) then
            L_o ← L_o ∪ mp_i
        end if
    end for
end for
```

Algorithm 2 Finding all (t_o, t_r) pairs in intervals of length up to t

Output: List of (t_o, t_r) pairs L_r

```
for every i → 1, ..., |W| do
    for every tuple of consecutive t_o^i, t_o^j ∈ {L_o ∪ t} do
        if ∃t_r with t_o^i < t_r < t_o^j which satisfies (8) then
            L_r ← L_r ∪ ⟨t_o, t_r⟩
        end if
    end for
end for
```

Example 2: Continuing from Example 1, three overload points (t_o) and three recovery points (t_r) exist in all time intervals t of length $0 < t \leq (\text{LCM}(6, 12) + 2(3-1))$ as defined in Theorem 2. The overload and associated recovery points up to the hyperperiod are: $(12, 14)$. Hence, the worst-case delay is two units. Figure 4 shows a tuple of an overload and a recovery point where the worst-case delay occurs.

A system enters into *continuous overload* if $\exists t_o : \forall t > t_o$ *sbf*$(t) <$ *dbf*(t). If the resource utilization is less than the workload utilization, the system will eventually experience continuous overload.

C. Schedulability Analysis with Overloads

Schedulability analysis is one of the key requirements in real-time systems. A hard real-time system will be schedulable if *sbf*$(t) \geq$ *dbf*(t) at any time interval t. However, the schedulability condition *sbf*$(t) \geq$ *dbf*(t) in any time interval t is not applicable for soft real-time system that can tolerate overloads. Therefore in the following, the schedulability analysis condition for *EDF* is defined in the presence of overloads (Theorem 3), characterized by the maximum tolerable delay.

Theorem 3: Given a system workload $W = \{T_1, T_2, \ldots, T_n\}$ with tolerable δ^* and a given resource model, W is schedulable if and only if the resource demand in any time interval exceeds the resource supply during the same time interval for no more than δ^* consecutive units of time. Furthermore, this only has to be checked for time intervals up to length $\text{LCM}(p_1, \ldots, p_n, \lambda) + 2(\lambda - \theta)$.

Theorem 2 establishes the proof of Theorem 3, because $f(t) =$ *sbf*$(t) -$ *dbf*(t) repeats after $\text{LCM}(p_1, \ldots, p_n, \lambda) + 2(\lambda - \theta)$.

V. Finding an Efficient Resource Supply

For a given system specification consisting of a workload and a worst-case delay, the objective of the developer is to provision the system with an applicable resource supply. In [28], the authors show how to calculate θ under the *EDF* scheduling policy with a given demand and resource period λ. Since our approach permits overloads, the technique specified in [28] is inapplicable. Furthermore, our target is to find θ without a predefined λ.

Since many possible resource supplies exist for a given workload, our method of calculating an efficient resource supply uses a cost function to choose one resource supply among many. The resource period λ and the supply θ are the parameters of the cost function. Our approach not only focuses on increasing the system throughput by lowering the resource usage (corresponding to a small θ/λ), but also reducing the number of context switches (corresponding to a large λ). A larger λ will decrease the number of context switches because the resource accounting mechanism in the operating system will preempt the workload less frequently.

Assuming tasks with periods equal to deadlines, periodic transient overloads ($U_W = U_R$), and a periodic resource model, we propose an efficient periodic resource supply model and calculate λ and θ using the following lines:

- the diagonal lines (e.g., lines 0, 1, 2, and 3 in Figure 5) that pass through the points where *sbf*(t) increases,
- the horizontal lines (e.g., lines 4, 5, and 6 in Figure 5) that pass through the *dbf*(t) where *dbf*$(t) > 0$,
- the vertical lines (e.g., lines 7, 8, and 9 in Figure 5) that pass through the points where *dbf*(t) might equal to *sbf*(t) after an overload occurred at $t - \delta^*$.

The diagonal, horizontal, and vertical lines intersect (as shown in Figure 5) for a certain λ and the corresponding θ. The calculation of the lines of interest is as follows.

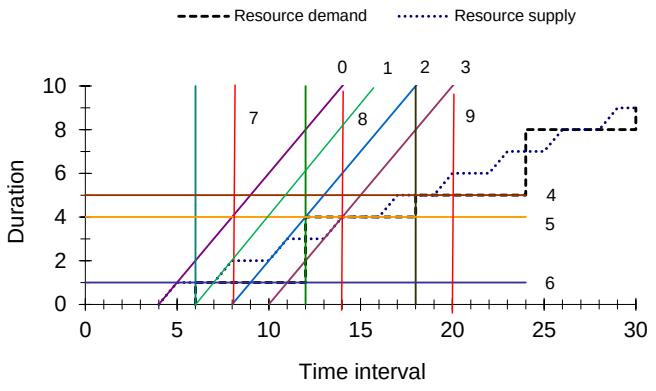

Fig. 5. Finding intersection points for a given *dbf* and δ^*

Calculating diagonal lines: The diagonal lines as shown in Figure 5 intercept the points where *sbf*(t) increases at periodic intervals after an initial time-interval offset of $2(\lambda - \theta)$. The *sbf* increases by θ in each time interval of length λ after the initial offset. At each $t = 2(\lambda-\theta)+k(\lambda-\theta), k \in \mathbb{N}$, the slope of *sbf* (t) is 1. The diagonal lines are of interest because they represent all the points where the *sbf* increases. Equation 11 represents the set of all diagonal lines.

$$\{y_k(t) = (t - (k+2)(\lambda - \theta)), k \in \mathbb{N}\}. \tag{11}$$

Example 3: Using the workload $W = \{T_1(6,1), T_2(12,2)\}$ presented in Example 1, the tolerable worst-case delay $\delta^* = 2$, and Equation 11, the following equations correspond to the four diagonal lines shown in Figure 5 such that $k = 0, \ldots, 3$.

$$y_0(t) = t - 2(\lambda - \theta), \tag{12}$$
$$y_1(t) = t - 3(\lambda - \theta), \tag{13}$$
$$y_2(t) = t - 4(\lambda - \theta), \tag{14}$$
$$y_3(t) = t - 5(\lambda - \theta). \tag{15}$$

Calculating horizontal lines: The horizontal lines shown in Figure 5 intercept the y-axis at the sum of the execution time units of a currently executing instance of a task and the execution time of all the preceding periodic instances of the current task and higher priority tasks. The y-intercept points are from the horizontal lines drawn on the *dbf*(t) such that *dbf*$(t) > 0$. The horizontal lines are of interest because they contain the points where *sbf*(t) may be equal to *dbf*(t).

To devise an equation for representing the horizontal lines that intercept the y-axis, we assume a vector v containing the execution times of all tasks (i.e., $v = (e_1, e_2, \ldots, e_n)$). We also define C as a set of indices that refer to the possible number of instances of a task.

$$C = \left\{ (\alpha_0, \alpha_1, \ldots, \alpha_n) \mid \exists t \in \mathbb{R}^+ \; \forall T_i \in W : \alpha_i = \left\lfloor \frac{t}{p_i} \right\rfloor \right\}. \tag{16}$$

Equation 17 represents the set of horizontal lines that originate from y-intercept points.

$$D = \left\{ y \mid y = \sum \alpha v^T, \alpha \in C \right\}. \tag{17}$$

(Continuing Example 3). In the following, the horizontal lines of Figure 5 represent the first few lines drawn at the y-intercept points using Equation 17 for Example 3 (all the lines are indexed numerically in ascending order).

$$y_0 = e_1,$$
$$y_1 = 2e_1 + e_2,$$
$$y_2 = 3e_1 + e_2.$$

Calculating vertical lines: Finally, the solid vertical lines shown in Figure 5 intercept the x-axis at the positive integer multiples of p_i of each task T_i. Since the time intervals p_i represent the time intervals where an overload might have occurred, the recovery points will be located on the right of overload points by an amount of δ^*. Therefore, these vertical lines will be shifted right by an amount of the worst-case delay

δ^* (e.g., lines 7 , 8, and 9) that may pass through the recovery points where $sbf(t) = dbf(t)$ when there is an overload at time interval $t - \delta^*$.

The *dbf* increases at time intervals of length $p_i\omega$ where $\omega \in \mathbb{N}^+$ for $T_i \in W$. Equation 18 represents the set of the vertical lines where $sbf(t) = dbf(t)$.

$$S = \left\{x \mid x = p_i\omega + \delta^*, T_i \in W, \omega \in \mathbb{N}^+\right\}. \tag{18}$$

(Continuing Example 3). In the following, the horizontal lines represent the first few lines drawn at the x-intercept points where $sbf(t)$ might equal to $dbf(t)$ using Equation 18 for Example 3.

$$\begin{aligned} x_0 &= 8, \\ x_1 &= 14, \\ x_2 &= 20. \end{aligned}$$

To find λ and θ, our method uses the intersection points resulting from these equations. First, we solve Equation 11 with Equation 17, and then replace θ and t using $\sum \frac{e_i}{p_i} = \frac{\theta}{\lambda}$ (i.e., $U_W = U_R$) and Equation 18. This yields

$$\begin{aligned} \lambda &= \theta - \left(\frac{\sum \alpha v^T - t}{(k+2)}\right) \\ &= \sum \frac{e_i}{p_i}\lambda - \left(\frac{\sum \alpha v^T - t}{(k+2)}\right) \\ &= \sum \frac{e_i}{p_i}\lambda - \left(\frac{\sum \alpha v^T - p_i\omega_i - \delta^*}{(k+2)}\right) \end{aligned}$$

$$\therefore \lambda = \frac{\sum \alpha v^T - p_i\omega_i - \delta^*}{(k+2)(\sum \frac{e_i}{p_i} - 1)} \tag{19}$$

(Continuing Example 3). By combining Equations $12 - 15$ with Equation 17 and replacing $\theta = \frac{1}{3}\lambda$, we get the following equations which later are replaced by Equation 18 to deduce λ and θ.

$$\begin{aligned} \lambda &= \frac{3}{4}(t - \sum \alpha v^T) && \text{(using Equation 12),} \\ \lambda &= \frac{1}{2}(t - \sum \alpha v^T) && \text{(using Equation 13),} \\ \lambda &= \frac{3}{8}(t - \sum \alpha v^T) && \text{(using Equation 14),} \\ \lambda &= \frac{3}{10}(t - \sum \alpha v^T) && \text{(using Equation 15).} \end{aligned}$$

Using the possible values of ω, and α until the hyperperiod (i.e., LCM$(p_1, \ldots, p_n, \lambda)$), we will get a set of (λ, θ) such that $\theta = \frac{1}{3}(\lambda)$. Using these assignments we can calculate δ^* using Algorithms **??** and 2, and check for the validity of the resource supply with respect to the workload demand. Thus the algorithm based on our proposed resource supply calculation model finds a list of resource supplies that allow the worst-case delay to be less than or equal to the value δ^* and chooses the best solution as $(\lambda, \theta) = (3, 1)$.

Since searching up to LCM$(p_1, \ldots, p_n, \lambda) + 2(\lambda - \theta)$ for each possible λ is too time consuming, we devise a time-efficient algorithm to reduce the search space to LCM$(p_1, \ldots, p_n) + 2(\lambda - \theta)$ and check the schedulability for $0 \leq t \leq$ LCM$(p_1, \ldots, p_n, \lambda) + 2(\lambda - \theta)$. If the identified supply leads to a system that is not schedulable until LCM$(p_1, \ldots, p_n) + 2(\lambda - \theta)$, then our method gradually increases the utilization of the resource supply until the system will become schedulable. The algorithm we devised based on the resource supply calculation equation (Eq. 19) is not shown in the paper because of the page limitations.

To find an efficient resource supply, the workflow is as follows for a given set of tasks: the utilization of the resource supply is kept the same as the workload utilization. The algorithm based on the supply calculation model searches for resource supplies that have recovery points at $t = \delta^* + m \cdot p_i$ (i.e., time intervals that denote the recovery of overloads) and calculates a fitting resource period. However, a resource supply that contains the recovery point may still be unusable, because the supply might have a worse δ^* at a later or earlier part of the *sbf*. Therefore, the algorithm searches for different supplies and checks them based on the method described in Section IV-A. Figure V shows multiple resource supplies for the Example 1 workload with $\delta^* = 1$; e.g., $\lambda = 1.5$ (blue color line in Figure V) and $\lambda = 1.12$ (red color line in Figure V). Our framework then selects from these candidates the one that has the largest λ to reduce the number of context switches.

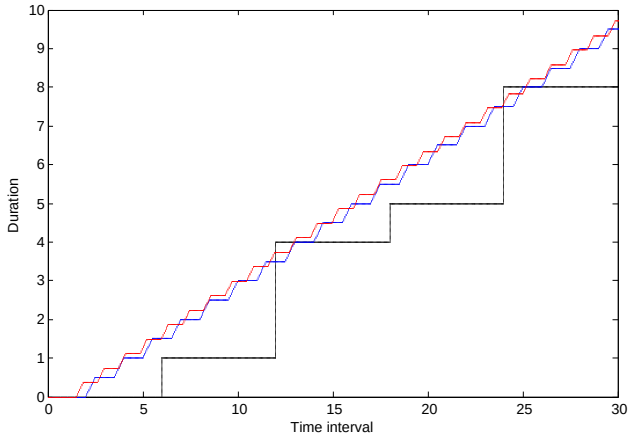

Fig. 6. Two of the candidate solutions for Example 1 with input delay *one*

Theorem 4 (Completeness): Any algorithm based on the proposed resource supply calculation model (Eq. 11, 17, and 18) and the proposed method to find a valid resource supply is complete.

Proof: In the proposed resource supply calculation model we see that the denominator of Equation 19 increases by the value $k \in \mathbb{N}$, because the utilization of the workload is constant. Therefore, the supply calculation model proceeds by

decreasing λ, which automatically decreases θ as $U_W = U_R$. With a small enough λ and θ, the *sbf* will become similar to a line with slope such that the δ^* constraint is preserved. The proposed model will always find a valid resource supply, if one exists for the given inputs. Therefore, an algorithm based on the model is complete. ■

Note that the secondary goal of maximizing λ is irrelevant for completeness (c.f., Theorem 4).

Corollary 1 (Soundness): Any complete algorithm based on the proposed model (Eq. 11, 17, and 18) and the proposed method to find a valid resource supply is sound.

Proof: Since a complete algorithm based on the proposed model checks the validity of the resource supply after finding one and eventually finds at least a valid solution, the algorithm based on the model is sound. ■

Feasibility analysis refers to whether a task set is feasible under a resource model. Feasibility ensures that there exists a resource supply that can satisfy the requirements of the tasks. Theorem 5 denotes that the mathematical model we use to calculate an efficient resource supply is feasible under a periodic resource model for workloads with bounded overloads.

Theorem 5: Given a system workload $W = \{T_1, T_2, \ldots, T_n\}$ with tolerable δ^*, the proposed model is feasible if and only if there exists a resource supply such that the resource demand in any time interval exceeds the resource supply during the same time interval for no more than δ^* consecutive units of time.

Proof: Theorem 4 establishes the proof of Theorem 5, because there exist always a resource supply such that maximum delay is bounded by δ^*. ■

VI. Experimental Analysis of a Control System

We developed a MATLAB-based application called *sbFinder* based on the results shown in this work. To demonstrate the utility of our technique for designing the resource supply in the context of a control system, we consider the problem of simultaneously stabilizing two plants with a single computational resource. The first plant, denoted by Σ_1, is an inverted pendulum mounted on a cart, and is given by the following linearized dynamical system [19]:

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.1818 & 2.6727 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.4545 & 31.1818 & 0 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ 1.8182 \\ 0 \\ 4.5455 \end{bmatrix}}_{B} u(t). \tag{20}$$

The state feedback controller for this system is designed (under nominal delay-free conditions) as

$$u(t) = \underbrace{\begin{bmatrix} 2.6939 & 3.5571 & -23.5576 & -4.462 \end{bmatrix}}_{K} x(t).$$

The plant is sampled every $p_1 = 0.02$ seconds. Using Theorem 1, the maximum value of η is 0.0732. The worst-case delay in this system then is $\delta_1 = \eta - 2p_1 = 0.032$ seconds.

The second plant, denoted Σ_2, is a chemical distillation column modeled as a linearized system with 8 states and 4 inputs; the exact model can be found in [29], and we omit the details here to save space. The nominal state feedback controller is designed to place the closed loop poles at $-1, -1.1, -1.2, -1.3, -1.4, -1.5, -1.6, -1.7$, and this produces the control gain K. The plant is sampled every $p_2 = 0.15$ seconds. Once again applying Theorem 1, the maximum value of η is 0.3340, and thus the maximum delay for obtaining computational resources that can be tolerated by this control system is $\delta_2 = \eta - 2p_2 = 0.034$ seconds.

The computation of both control inputs is done by a single processor. Thus, to maintain stability, the processor must guarantee that the worst-case delay for calculating any control input is $\delta^* = \min(\delta_1, \delta_2) = 0.032$ seconds. We take our unit of time to be 0.01 seconds, and assume that the processor can compute the control input for either plant within this length of time. Thus, in terms of this time-unit, the period and execution time for the first plant are $p_1 = 2$ and $e_1 = 1$, respectively, and the period and execution time for the second plant are $p_2 = 15$ and $e_2 = 1$, respectively. The maximum tolerable delay δ^* for both systems is $\delta^* = 3$.

With this workload and delay specifications, our proposed model produces a list of 138 solutions that are valid. The list contains 132 different assignments of resource supply that have the minimum utilization. Traversing the list for the maximum resource period yields $\lambda = 2.5$, $\theta = 1.4167$, $\delta^* = 1.8$, and utilization $= 0.5667$.

VII. Related Work

Shin and Lee present schedulability analysis based on the *sbf* [26], [27] and the *dbf* [1], [17] for the compositional real-time scheduling framework. This framework can be used to establish global (system level) timing properties by composing individual timing properties. The authors present schedulability conditions for the standard Liu and Layland periodic resource task model and propose a periodic resource model under earliest deadline first (*EDF*) and rate monotonic (RM) scheduling that allows the composition of multiple timing requirements into a single timing requirement. In the related work [26], the authors analyze schedulability of a bounded delay resource partition model in terms of the *sbf* and the *dbf*. Deducing a single timing requirement out of multiple timing properties creates some new challenges which have been solved in a number of subsequent works [24], [27]. As a variant of the system model from [26], Shin *et al.* also propose algorithms that define optimal interfaces for the subsystems which may share resources. Integrating subsystems into a system having optimal interfaces motivates the development of adaptive and reconfigurable systems.

An important aspect of using the *sbf* and the *dbf* is the optimized use of the resources. Easwaran *et al.* [9] show that selecting a particular resource model that minimizes the collective resource requirements facilitate systems to change components on the fly. Lee *et al.* propose an optimization framework for maximizing the quality of service (QoS) under K random failures on schedulability. The authors use Langarian duality [16] for distributed computation that results in optimal solutions.

Mok, Feng, and Chen [21] introduce the concept of a supply function to measure the minimum amount of com-

puting resources provided to a static partition. Wandler and Thiele [30] propose the concept of interface-based design that uses real-time calculus and modular performance analysis to compute the supply curves. Lipari and Bini [18] derive a set of supply functions that are feasible to schedule an application. Later Bini et al. propose an optimization framework [3] to select the minimum bandwidth required of a EDF task set. These works use the fraction of computing resource supplied by the processor and the initial delay of the resource to ensure that minimum bandwidth is given to the workload demand, but do not consider delays due to the transient overloads that the tasks may tolerate and the existence of a periodic resource model (a special class of supply functions) that Shin and Lee [25] introduce.

Devi and Anderson [8] introduce tardiness bounds under global EDF scheduling on a multiprocessor for soft real-time systems. However, the tardiness bounds are not derived in terms of the supply and demand bound functions for a compositional framework as discussed in [26]. Kumar et al. [15] propose a model to compute the resource with a given delay bound from a stream of jobs characterized by an input arrival trace. However, the arrival jobs are not specified with a certain delay bound that we assume in this work to characterize application-specific tasks that can tolerate overloads or delays. Moreover, the delay is calculated in terms of time rather than the time intervals that we follow because we attempt to calculate the delay from the supply and demand bound as defined in [26], [27] which are functions of time intervals. Buttazzo et al. [10] introduce elastic scheduling that allows to vary the period of a task based on its flexibility specified in the task model. This model inherently allows to tolerate overloads to a certain amount but does not use the concept of supply and demand bound functions we use to compute an efficient periodic resource model towards building a compositional system. Hence our work is in-line with the other work in the literature but differs in finding an efficient resource model due to the time-interval analysis of supply and demand bound functions for systems that can tolerate bounded transient overloads.

VIII. Discussion

Compositionality: A compositional framework can combine different specifications and build up a schedule that satisfies the workload demands. The system calculates the most suitable resource supply for each of the specification's demands. Each resource supply turns into the workload demand while using the compositional framework. Earlier work on the periodic resource model [26] assumes no overloads or delays in the workload. Using our work to extend [26], it is possible to deal with workloads with bounded transient overloads. Definition 6 describes the composition method in the context of transient overloads.

Definition 6 (Composition method): Given a number of scheduling models $M_1 \dots M_n$, a compositional scheduling model $M_P(W_P, R_P, EDF)$ can be derived by mapping the resource model of a child scheduling model $R_i(\lambda_i, \theta_i)$ to its periodic task $T_i(p_i, e_i)$ and including any new tasks (T'_i) such that $W_P = \{T_1(\lambda_i, \theta_i), \dots, T_n(\lambda_n, \theta_n)\} \cup \{T'_i(p'_i, e'_i)\}$.

Handling sporadic tasks: With a slight modification, the proposed resource supply model can handle sporadic tasks. Sporadic tasks have a minimum inter-arrival time. Therefore, the *dbf* for sporadic tasks is different from periodic tasks. For sporadic tasks, $dbf(W, EDF) = \sum_{T_i \in W} \max(0, (\left\lfloor \frac{t-d_i}{p_i} \right\rfloor + 1)e_i)$. In the worst case, sporadic tasks can arrive periodically based on their inter-arrival time and our model can still handle such workloads.

Relation between overload duration and overload severity: The duration and the severity of an overload are related. During the work, we made the following observations:

1) A large overload severity implies a long overload duration and thus a long delay.
2) A long overload duration (and thus a long delay) does not necessarily imply a large overhead severity.

These two observations originate from the fact that the best possible resource supply is $R(1, 1)$ in which the system receives all resources. In such a scenario, the slope of the *sbf* is 1. Thus, the overload delay is always at least equal to the overload severity. Hence, a large overload severity implies a long overload duration.

On the other hand, the worst possible resource supply is $R(x, 1)$ where $x \to 0$. In this scenario, the slope of the *sbf* is close to 0. Thus, even a small overload severity can result in a long overload duration.

Multiprocessor systems: The way to deal with multiprocessor systems is to use partitioning algorithms [2]. *EDF* is not guaranteed to be optimal for multiprocessor systems, although it is optimal for uniprocessor systems. Therefore, prior to running the *EDF* scheduling policy for uniprocessor systems, a partitioning algorithm can distribute the tasks of multiprocessors into several uniprocessors.

Hyperperiods: The hyperperiod grows exponentially as a function of the longest period, number of tasks, and co-primeness of the period of the tasks. Task period selection to minimize the hyperperiod is an interesting problem and has been studied in previous work [5].

IX. Conclusion

This paper presents a holistic analysis to characterize overloads using overload points and recovery points for systems experiencing transient overloads. To understand the impact of overloads, we define overload metrics such as the worst-case delay and the worst-case severity. Using the analysis of overloads, we propose an efficient resource supply model for a given workload and a tolerable worst-case delay. Control engineers can use the framework for feedback control systems, which is demonstrated by simultaneously stabilizing two plants with a single computational resource.

Acknowledgements.

This research was supported in part by NSERC DG 357121-2008, ORF-RE03-045, ORF-RE04-036, ORF-RE04-039, APCPJ 386797-09, CFI 20314 and CMC, NSERC DG-114741-2010, and the industrial partners associated with these projects.

REFERENCES

[1] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized Multiframe Tasks. *Real-Time Systems*, 17(1):5–22, 1999.

[2] S. Baruah and N. Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Transactions on Computers*, 55, 2006.

[3] Enrico Bini, Giorgio Buttazzo, and Yifan Wu. Selecting the minimum consumed bandwidth of an EDF task set, 2009.

[4] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] V. Brocal, P. Balbastre, and R. Ballester. Task period selection to minimize hyperperiod. 2011.

[6] C.T. Chen. *Linear Systems, Theory and Design*. Oxford University Press, 1999.

[7] Q.-L. Han D. Yue and C. Peng. State feedback controller design for networked control systems. *IEEE Transactions on Circuits and Systems – II: Express Briefs*, 51(11):640–644, Nov. 2004.

[8] U.M.C Devi and J.H Anderson. Tardiness Bounds under Global EDF Scheduling on a Multiprocessor. In *Real-Time Systems Symposium*, dec. 2005.

[9] A. Easwaran, I. Shin, O. Sokolsky, and I. Lee. Incremental schedulability analysis of hierarchical real-time components. In *R. Shelton, ORMSC, Open Engineering*, pages 272–281. ACM Press, 2006.

[10] G. C. Buttazzo and M. Caccamo and L. Abeni. Elastic Scheduling for Flexible Workload Management. *IEEE Transactions on Computers*, 51:289–302, 2002.

[11] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx/, April 2011.

[12] V. Gupta, A. F. Dana, J. Hespanha, R. M. Murray, and B. Hassibi. Data transmission over networks for estimation and control. *IEEE Transactions on Automatic Control*, 54(8):1807–1819, Aug. 2009.

[13] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138 – 162, Jan. 2007.

[14] O. C. Imer, S. Yuksel, and T. Basar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 42(9):1429–1439, Sep. 2006.

[15] P. Kumar, J. Chen, L. Thiele, A. Schranzhofer, and G.C. Buttazzo. Real-time analysis of servers for general job arrivals. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, volume 1, pages 251 –258, aug. 2011.

[16] J. Lee, I. Shin, and A. Easwaran. Online robust optimization framework for qos guarantees in distributed soft real-time systems. In *Proceedings of the tenth ACM international conference on Embedded software*, EMSOFT, pages 89–98. ACM, 2010.

[17] J. P. Lehoczky, L. Sha, and J. K. Strosnider. Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. In *IEEE Real-Time Systems Symposium*, pages 261–270, 1987.

[18] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *15th Euromicro Conference on Real-Time Systems*, pages 151–158, 2003.

[19] B. Messner and D. Tilbury. Control tutorials for MATLAB and Simulink. http://www.engin.umich.edu/group/ctm/examples/pend/invpen.html.

[20] A. Mok. Firm real-time systems. *ACM Computing Surveys*, 28, 1996.

[21] A. K. Mok and A. Feng. Towards compositionality in real-time resource partitioning based on regularity bounds. In *IEEE Real-Time Systems Symposium*, pages 129–138, 2001.

[22] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam. The wireless control network: A new approach for control over networks. *IEEE Transactions on Automatic Control*, 56(10):2305–2318, Oct. 2011.

[23] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. Foundations of control and estimation over lossy networks. *Proc. of the IEEE*, 95:163–187, 2007.

[24] I. Shin, M. Behnam, T. Nolte, and M. Nolin. Synthesis of optimal interfaces for hierarchical scheduling with resources. In *IEEE Real-Time Systems Symposium*, pages 209–220, 2008.

[25] I. Shin and I. Lee. Periodic Resource Model for Compositional Real-time Guarantees. In *Real-Time Systems Symposium*, pages 2–13, 2003.

[26] I. Shin and I. Lee. Compositional real-time scheduling framework. In *IEEE Real-Time Systems Symposium*, pages 57–67, 2004.

[27] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Trans. Embed. Comput. Syst.*, 7:30:1–30:39, May 2008.

[28] I. Shin and I. Lee. Periodic Resource Model for Compositional Real-time Guarantees. In *Technical Report*, pages 21– 15, 2010.

[29] S. Skogestad and I. Postlewaite. *Multivariable feedback control*. Wiley, 1996.

[30] E. Wandeler and L. Thiele. Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling. In *Proceedings of the 5th ACM international conference on Embedded software*, EMSOFT, 2005.

[31] W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21(1):84– 99, Feb 2001.