Multiprocessor Extensions to Real-Time Calculus

Hennadiy Leontyev*, Samarjit Chakraborty† and James H. Anderson*
*Department of Computer Science,University of North Carolina at Chapel Hill
{leontyev,anderson}@cs.unc.edu
†Institute for Real-Time Computer Systems (RCS), TU Munich
samarjit@tum.de

***Abstract*—Many embedded platforms consist of a heterogeneous collection of processing elements, memory modules, and communication subsystems. These components often implement different scheduling/arbitration policies, have different interfaces, and are supplied by different vendors. Hence, compositional techniques for modeling and analyzing such platforms are of interest. In prior work, the real-time calculus framework has proven to be very effective in this regard. However, real-time calculus has heretofore been limited to systems with uniprocessor processing elements, which is a serious impediment given the advent of multicore technologies. In this paper, a two-step approach is proposed that allows the power of real-time calculus to be applied in globally-scheduled multiprocessor systems: first, assuming that job response-time bounds are given, determine whether these bounds are met; second, using these bounds, determine the resulting residual processor supply and streams of job completion events using formalisms from real-time calculus. For this methodology to be applied in settings where response-time bounds are not specified, such bounds must be determined. Though this is an issue that warrants further investigation, a method is discussed for calculating such bounds that is applicable to a large family of fixed job-priority schedulers. The utility of the proposed analysis framework is demonstrated using a case study.**

***Keywords*-component-based design; multiprocessor scheduling; real-time calculus**

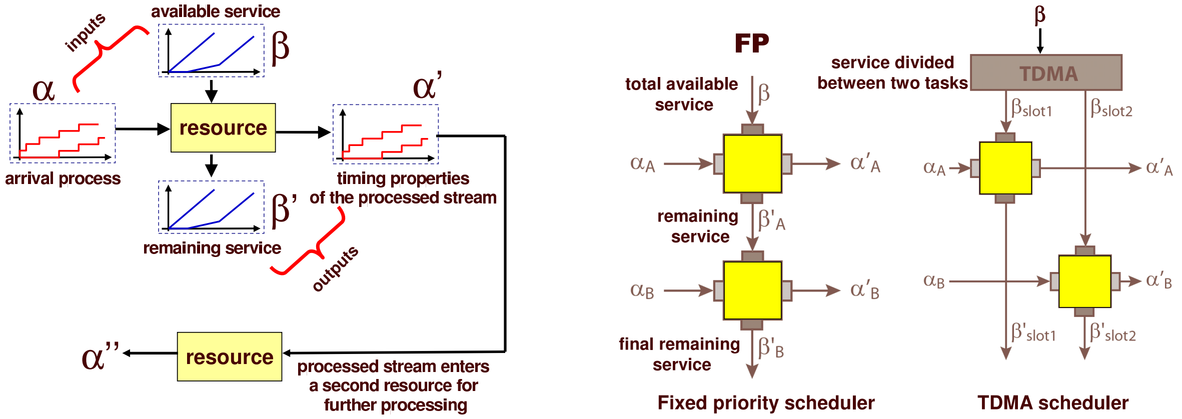
I. INTRODUCTION

The increasing complexity and heterogeneity of modern embedded platforms have led to growing interest in compositional modeling and analysis techniques [15]. In devising such techniques, the goal is not only to analyze the individual components of a platform in isolation, but also to compose different analysis results to estimate the timing and performance characteristics of the entire platform. Such analysis should be applicable even if individual processing and communication elements implement different scheduling/arbitration policies, have different interfaces, and are supplied by different vendors. These complicating factors often cause standard event models (e.g., periodic, sporadic, etc.) and schedulability-analysis techniques to lead to overly pessimistic results or to be altogether inapplicable.

To overcome this difficulty, a compositional framework — often referred to as *real-time calculus* — was proposed by Chakraborty et al. in [3] and then subsequently extended in a number of papers (e.g., see [4]). Real-time calculus is a specialization of *network calculus*, which was proposed by Cruz in 1991 [5], [6] and has been widely used to analyze communication networks since then. Real-time calculus specializes network calculus to the domain of real-time and embedded systems by, for example, adding techniques to model different schedulers and mode/state-based information (e.g., see [14]). A number of schedulability tests have also been derived based upon network calculus. An overview of these tests can be found in [18].

In real-time calculus, timing properties of event streams are represented using upper and lower bounds on the number of events that can arrive over any time interval of a specified length. These bounds are given by functions $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$, which specify the maximum and minimum number of events, respectively, that can arrive at a processing/communication resource within any time interval of length Δ (or the maximum/minimum number of possible task activations within any Δ). The service offered by a resource is similarly specified using functions $\beta^u(\Delta)$ and $\beta^l(\Delta)$, which specify the maximum and minimum number of serviced events, respectively, within any interval of length Δ. Given the functions α^u and α^l corresponding to an event stream arriving at a resource, and the service β^u and β^l offered by it, it is possible to compute the timing properties of the processed stream and remaining processing capacity, i.e., functions $\alpha^{u\prime}$, $\alpha^{l\prime}$, $\beta^{u\prime}$, and $\beta^{l\prime}$, as illustrated in Fig. 1(a), as well as the maximum backlog and delay experienced by the stream. As shown in the same figure, the computed functions $\alpha^{u\prime}$ and $\alpha^{l\prime}$ can then serve as inputs to the next resource on which this stream is further processed. By repeating this procedure until all resources in the system have been considered, timing properties of the fully-processed stream can be determined, as well as the end-to-end event delay and total backlog. This forms the basis for composing the analysis for individual resources, to derive timing/performance results for the full system.

Similarly, for any resource with tasks being scheduled according to some scheduling policy, it is also possible to compute bounds ($\beta^u(\Delta)$ and $\beta^l(\Delta)$) on the service available to its individual tasks. Fig. 1(b) shows how this is done for the *fixed-priority* (FP) and *time-division-multiple-access* (TDMA) policies. As shown in this figure, for the FP policy, the *remaining* service after processing Stream A serves



(a) (b)

Figure 1. **(a)** Computing the timing properties of the processed stream using real-time calculus. **(b)** Scheduling networks for fixed priority and TDMA schedulers.

as the input (or, is available) to Stream B. On the other hand, for the TDMA policy, the total service β is split between the services available to the two streams. Similar so called *scheduling networks* [4] can be constructed for other scheduling policies as well. Various operations on the arrival and service curves α and β, as well as procedures for the analysis of scheduling networks on uniprocessors (and partitioned systems) have been implemented in the RTC (real-time calculus) toolbox [17], which is a MATLAB-based library that can be used for modeling and analyzing distributed real-time systems.

Our contribution. Unfortunately, none of the compositional techniques described above can be used when the resource in question is a multiprocessor that is scheduled using a *global* multiprocessor scheduling algorithm. In particular, when such algorithms are used, processors may be idle even though tasks are available for execution, as tasks must execute sequentially; this situation does not arise on uniprocessors and thus is not addressed in uniprocessor compositional techniques.

There are two reasons why existing compositional techniques need to be extended to incorporate such multiprocessors. First, multicore chips are becoming increasingly common. Second, viewing a multiprocessor system as a collection of independent uniprocessors and applying partitioning techniques is unnecessarily restrictive and precludes supporting workloads that fundamentally require global scheduling approaches (such a workload is considered in a case study presented later).

Motivated by these observations, we present in this paper an extension of the real-time calculus framework [3], [4] that incorporates globally-scheduled multiprocessors and is compatible with the RTC toolbox. The core of our framework is a pseudo-polynomial-time procedure that, given a collection of arrival curves for input streams α^u and α^l, their execution requirements, and the available resource supply, checks that event delays on such a multiprocessor reside within specified bounds. Second, using these event delays, we compute arrival curves for the processed streams, $\alpha^{u\prime}$ and $\alpha^{l\prime}$, and the remaining-total-service curve; these curves — as in the uniprocessor case — can in turn be used as input for other resources, thereby resulting in a compositional framework (as shown in Fig. 1(a)). To apply these results, per-stream delay bounds must be given. In settings where such bounds are not given, they must be determined. We present a simple method for calculating such bounds, but a comprehensive evaluation of its properties is deferred to future work.

Prior work. Our work is based upon multiprocessor schedulability tests by Baruah [1] and Leontyev and Anderson [9]. In some aspects, the presented analysis is also similar to results by Bertogna et al. [2], Shin et al. [16], and Zhang and Burns [19]. The main difference between our work and these prior efforts is that we consider more general task arrival and execution models, viz. those supported by the real-time calculus framework. Also, we consider the case when one or more processors can be partially available, which is similar to analysis in [16], where partial availability is considered in the context of hierarchical scheduling. Our work is different from that in [18] and related works listed there in that we are primarily concerned with multiprocessor scheduling and earliest-deadline-first-like algorithms. Due to space constraints, this paper covers only the most essential parts of the new framework. Proofs for some lemmas and claims are omitted here but can be found in [11].

The rest of the paper is organized as follows. Sec. II presents our task model. In Secs. III and IV, timing characteristics of processed streams and the remaining supply are computed. In Secs. V and VI, the response-time-bound test is presented and its time complexity is discussed. In Sec. VII, we present closed-form expressions for calculating response-time bounds. Sec. VIII presents a case study for our analysis, and finally, Sec. IX discusses some directions for future work.

II. Task Model

In this paper, we consider a task set $\tau = \{T_1, \ldots, T_n\}$. Each task has incoming jobs that are processed by a multiprocessor consisting of $m \geq 2$ unit-speed processors. We assume that $n \geq m$. We also assume that all time quantities are integral.

The j^{th} job of T_i, where $j \geq 1$, is denoted $T_{i,j}$. The *arrival* (or *release*) *time* of $T_{i,j}$ is denoted $r_{i,j}$. The *completion time* of $T_{i,j}$ is denoted $f_{i,j}$ and the delay between its start time and completion, $f_{i,j} - r_{i,j}$, is called its *response time*. As in prior work on real-time calculus, we wish to be able to accommodate very general assumptions concerning job executions and arrivals and the available service. Most of the remaining definitions in this section are devoted to formalizing the assumptions we require. Table I summarizes the notation introduced in this section.

Definition 1. $\gamma_i^u(k)$ ($\gamma_i^l(k)$) denotes an upper (lower) bound on the total execution time of any k consecutive jobs of T_i. (We assume $\gamma_i^u(k) = 0$ for all $k \leq 0$ and $\gamma_i^u(k) \leq \gamma_i^u(k+1)$, and similarly for $\gamma_i^l(k)$.) These definitions are equivalent to the workload demand curves in [12].

Example 1. Suppose that task T_i's job execution times follow a pattern $1, 5, 2, 1, 5, 2, \ldots$. Then, $\gamma_i^u(1) = 5$, $\gamma_i^u(2) = 7$, $\gamma_i^u(3) = 8$, $\gamma_i^u(4) = 13$, etc. Also, $\gamma_i^l(1) = 1$, $\gamma_i^l(2) = 3$, $\gamma_i^l(3) = 8$, $\gamma_i^l(4) = 9$, etc.

Definition 2. The *arrival function* $\alpha_i^u(\Delta)$ ($\alpha_i^l(\Delta)$) provides an upper (lower) bound on the number of jobs of T_i that can arrive within *any* time interval $(x, x+\Delta]$, where $x \geq 0$ and $\Delta > 0$ [4]. (We assume $\alpha_i^u(\Delta) = 0$ for all $\Delta \leq 0$.) $\alpha_i(\Delta)$ denotes the pair $(\alpha_i^u(\Delta), \alpha_i^l(\Delta))$.

Example 2. The widely-studied periodic and sporadic task models are subcases of this more general task model. In both models, consecutive job arrivals of T_i are separated by at least p_i time units, where p_i is the *period* of T_i, and each job requires at most $e_i^{\max}$ execution units. Therefore, under both models, $\alpha_i^u(\Delta) = \left\lceil \frac{\Delta}{p_i} \right\rceil$ and $\gamma_i^u(k) = k \cdot e_i^{\max}$.

Definition 3. Let $\mathcal{A}_i^{-1}(k) = \inf\{\Delta \mid \alpha_i^u(\Delta) > k\}$, where $\Delta > 0$. This function characterizes the minimum length of the time interval $(x, x+\Delta]$ during which jobs $T_{i,j+1}, \ldots, T_{i,j+k}$ can be released for some j, assuming $T_{i,j}$ is released at time x. We define $\mathcal{A}_i^{-1}(0) = 0$ and require that there exists $K_i \geq 1$ such that

$$\mathcal{A}_i^{-1}(K_i) \geq \gamma_i^u(K_i). \tag{1}$$

We further require that there exists $R_i > 0$ and $B_i \geq 0$, where $R_i = \lim_{\Delta \to +\infty} \frac{\alpha_i^u(\Delta)}{\Delta}$, such that

$$\alpha_i^u(\Delta) \leq R_i \cdot \Delta + B_i \text{ for all } \Delta \geq 0. \tag{2}$$

Also, we assume that there exists $\overline{e_i} > 0$ and v_i, where $\overline{e_i} = \lim_{k \to +\infty} \frac{\gamma_i^u(k)}{k}$, such that

$$\gamma_i^u(k) \leq \overline{e_i} \cdot k + v_i \text{ for all } k \geq 1. \tag{3}$$

(1) is needed in order to prevent task T_i from overloading the system. In (2), R_i characterizes the long-term arrival rate of task T_i's jobs and B_i characterizes the degree of burstiness of the arrival sequence. In (3), the parameter $\overline{e_i}$ denotes the average worst-case job execution time of T_i.

Definition 4. Let $u_i = R_i \cdot \overline{e_i}$. This quantity denotes the average long-term utilization of task T_i. We require that $0 < u_i \leq 1$. Let $U_{sum} = \sum_{T_i \in \tau} u_i$.

Example 3. Under the sporadic task model, $R_i = \lim_{\Delta \to +\infty} \left(\left\lfloor \frac{\Delta}{p_i} \right\rfloor + 1 \right) / \Delta = \frac{1}{p_i}$ and $\overline{e_i} = e_i^{\max}$, so $u_i = R_i \cdot \overline{e_i} = \frac{e_i^{\max}}{p_i}$.

Definition 5. Let $\mathsf{supply}_h(t, \Delta)$ be the total amount of processor time available to tasks in τ on processor h in the interval $[t, t+\Delta)$, where $\Delta \geq 0$. Let $\mathsf{Supply}(t, \Delta) = \sum_{h=1}^{m} \mathsf{supply}_h(t, \Delta)$ be the cumulative processor supply in

Table I
Model notation.

Input parameters	
$\alpha_i^u(\Delta)$ ($\alpha_i^l(\Delta)$)	Max. (min.) number of job arrivals of T_i over Δ
$\gamma_i^u(k)$ ($\gamma_i^l(k)$)	Max. (min.) execution demand of any k consecutive jobs of T_i
$\mathcal{B}(\Delta)$	Min. guaranteed cumulative processor supply over Δ
Params. below can be found using RTC Toolbox	
$\widehat{U}$	Long-term avilable processor utilization
σ_{tot}	Maximum blackout time
F	The number of processors that are always available
$\mathcal{A}_i^{-1}(k)$	Pseudo-inverse of α_i^u
K_i	Min. integer s.t. $\mathcal{A}^{-1}(K_i) \geq \gamma_i^u(K_i)$
$\overline{e_i}$	T_i's average worst-case job execution time
v_i	Burstiness of the execution demand
R_i	Long-term arrival rate of T_i's jobs
B_i	Burstiness of the arrival curve
u_i	T_i's long-term utilization
U_{sum}	Total utilization
Θ_i below can be checked using the test in Sec. V	
Θ_i	T_i's response-time bound
Output calculated using the input and $\{\Theta_i\}$	
${\alpha^u}'_i(\Delta)$ (${\alpha^l}'_i(\Delta)$)	Max. (min.) number of job completions of T_i over Δ
$\mathcal{B}'(\Delta)$	Min. guaranteed unused processor supply over Δ

the interval $[t, t+\Delta)$.

Though we desire to make our analysis compatible with the real-time calculus framework, which requires that individual processor supplies be known, there exist many settings in which individual processor supply functions are not known and a lower bound on the cumulative available processor time is provided instead. (In uniprocessor real-time calculus, the available service is described as the number of incoming events processed by a PE during a time interval.) Note that if individual processor supply guarantees are known, a lower bound on the cumulative guaranteed supply can be computed easily.

Definition 6. Let $\mathcal{B}(\Delta) \leq \mathsf{Supply}(t, \Delta)$ be the guaranteed total time that all processors can provide to the tasks in τ during any time interval $[t, t+\Delta)$, where $\Delta \geq 0$. We assume that

$$\mathcal{B}(\Delta) \geq \max(0, \widehat{U} \cdot (\Delta - \sigma_{tot})), \tag{4}$$

where $\widehat{U} \in (0, m]$ and $\sigma_{tot} \geq 0$. We let F be the number of processors that are always available at any time. If all processors have unit speed, then $F = \max\{y \mid \forall \Delta \geq 0 :: \mathcal{B}(\Delta) \geq y \cdot \Delta\}$.

In the above definition, the parameters $\widehat{U}$, which is the total long-term fraction of processor time available to the tasks in τ on the entire platform, and σ_{tot}, which is the maximum duration of time when all processors are unavailable, are similar to those in the bounded delay model [13].

We require that (5) below holds for otherwise the system would be overloaded and job response times could be unbounded.

$$U_{sum} \leq \widehat{U} \tag{5}$$

We assume that released jobs are placed into a single global ready queue. When choosing a new job to schedule, the scheduler selects (and dequeues) the ready job of highest priority. An unfinished job is *pending* if it is released. A pending job is *ready* if its predecessor (if any) has completed execution. Note that the jobs of each task execute sequentially. Job priorities are determined as follows.

Definition 7. (prioritization rules) Associated with each job $T_{i,j}$ is a constant value $\chi_{i,j}$. If $\chi_{i,j} < \chi_{k,h}$ or $\chi_{i,j} = \chi_{k,h} \wedge (i < k \vee (i = k \wedge j < h))$, then the priority of $T_{i,j}$ is higher than that of $T_{k,h}$, denoted $T_{i,j} \prec T_{k,h}$. Additionally, we assume $j < h$ implies $\chi_{i,j} \leq \chi_{i,h}$ for each task T_i.

Example 4. Global earliest-deadline-first (GEDF) priorities can be defined by setting $\chi_{i,j} = r_{i,j} + D_i$ for each job $T_{i,j}$, where D_i is T_i's relative deadline. Global first-in-first-out (FIFO) priorities can be defined by setting $\chi_{i,j} = r_{i,j}$ [8].

The technical contributions of this paper are twofold. First, given per-task bounds on maximum job response times, we characterize the sequence of job completion events for each task T_i in terms of the next-stage arrival functions $\alpha_i^{u\prime}$ and $\alpha_i^{l\prime}$, and the remaining processor supply $\mathcal{B}'(\Delta)$; these, in turn, can serve as inputs to subsequent PEs, thereby resulting in a compositional technique.

Second, given a task set $\tau = \{T_1, \ldots, T_n\}$ and a multiprocessor platform characterized by a cumulative guaranteed processor time $\mathcal{B}(\Delta)$, we develop a sufficient test that verifies whether the maximum job response time of a task $T_i \in \tau$, $\max_j(f_{i,j} - r_{i,j})$, is at most Θ_i, where

$$\Theta_i \geq \max_{j \geq 1}(\gamma_i^u(j) - \mathcal{A}_i^{-1}(j-1)). \tag{6}$$

(It can be shown that the maximum job response time of T_i cannot be less than the right-hand-side of (6). Intuitively, $\gamma_i^u(j)$ is the maximum execution requirement of j consecutive jobs $T_{i,a}, \ldots, T_{i,a+j-1}$ and $\mathcal{A}_i^{-1}(j-1)$ is the minimum length of the interval where jobs $T_{i,a+1}, \ldots, T_{i,a+j-1}$ are released.) If Θ_i equals the relative deadline of a job, then the test will check whether the system is hard-real-time schedulable. Alternatively, if deadlines are allowed to be missed and Θ_i includes the maximum allowed deadline tardiness, then the test will check soft-real-time schedulability. Such a test allows workloads to be considered that fundamentally require global scheduling approaches. Unknown response-time bounds can be calculated by using closed-form expressions given in Sec. VII to determine initial bounds, and by then iteratively decreasing these bounds and applying the presented test to determine whether such decreased bounds are valid.

III. CALCULATING $\alpha_i^{u\prime}$ AND $\alpha_i^{l\prime}$

Let $\alpha_i^{u\prime}(\Delta)$ ($\alpha_i^{l\prime}(\Delta)$) be the maximum (respectively, minimum) number of job completions of task T_i over an interval $(x, x+\Delta]$, where $x \geq 0$. Bounds on these functions can be computed as follows.

Theorem 1. *If the response time of any job of T_i is at most Θ_i, then* $\alpha_i^{u\prime}(\Delta) \leq \min\left(\left\lceil \frac{\Delta}{\gamma_i^l(1)} \right\rceil,\ \alpha_i^u(\Delta + \Theta_i - \gamma_i^l(1))\right)$ *and* $\alpha_i^{l\prime}(\Delta) \geq \alpha_i^l(\Delta - \Theta_i + \gamma_i^l(1))$.

Proof: We prove the first inequality, leaving the second one to the reader. Consider an interval $(t_1, t_2]$ such that at least one job of T_i completes within it and $t_2 - t_1 = \Delta$. Let N_1, (N_2) be the index of the first (last) job of T_i completed within $(t_1, t_2]$. Then,

$$f_{i,N_1} > t_1 \quad \text{and} \quad f_{i,N_2} \leq t_2. \tag{7}$$

By the condition of the theorem, job $T_{i,j}$'s response time $f_{i,j} - r_{i,j}$ is at most Θ_i. By the definition of response time and Def. 1, $f_{i,j} - r_{i,j}$ is at least $\gamma_i^l(1)$. From (7), we thus have $r_{i,N_1} > t_1 - \Theta_i$ and $r_{i,N_2} \leq t_2 - \gamma_i^l(1)$. Thus, the number of jobs completed within the interval $(t_1, t_2]$, $N_2 - N_1 + 1$, is at most the number of jobs released within the interval $(t_1 - \Theta_i, t_2 - \gamma_i^l(1)]$. By Def. 2, we have $N_2 - N_1 + 1 \leq \alpha_i^u(t_2 - \gamma_i^l(1) - t_1 + \Theta_i) = \alpha_i^u(\Delta + \Theta_i - \gamma_i^l(1))$. If job $T_{i,j}$

completes at time $f_{i,j}$, then $T_{i,j+1}$ cannot complete earlier than $f_{i,j} + \gamma_i^l(1)$. Thus, job completions are separated by at least $\gamma_i^l(1)$ time units, and hence, at most $\left\lceil \frac{\Delta}{\gamma_i^l(1)} \right\rceil$ jobs can be completed within any interval of length Δ. ∎

IV. Calculating $\mathcal{B}'(\Delta)$

We now calculate a lower bound $\mathcal{B}'(\Delta)$ on processor time that is available after scheduling tasks $T_1, \ldots, T_n$. We first upper-bound the total allocation of jobs of T_i over any interval of length Δ.

Definition 8. Let $\mathsf{A}(T_i, I)$ be the total amount of time for which jobs of task T_i execute within the set of intervals I.

Lemma 1. *If the response time of any job of T_i is at most Θ_i, then $\mathsf{A}(T_i, [t, t+\Delta)) \leq \min(\Delta, \gamma_i^u(\alpha_i^u(\Delta + \Theta_i)))$.*

Proof: Consider an interval $[t, t+\Delta)$. The condition of the lemma implies that all of T_i's jobs released at or before time $t - \Theta_i$ complete by time t. Thus, the allocation of T_i within $[t, t+\Delta)$, $\mathsf{A}(T_i, [t, t+\Delta))$ is upper-bounded by the maximum execution demand of T_i's jobs released within the interval $(t - \Theta_i, t+\Delta]$. By Def. 2, there are at most $\alpha_i^u(\Delta + \Theta_i)$ jobs released within $(t - \Theta_i, t + \Delta]$, and by Def. 1, their total execution demand is at most $\gamma_i^u(\alpha_i^u(\Delta + \Theta_i))$. We thus have $\mathsf{A}(T_i, [t, t+\Delta)) \leq \gamma_i^u(\alpha_i^u(\Delta + \Theta_i))$. Also, $\mathsf{A}(T_i, [t, t+\Delta))$ cannot exceed the length of the interval $[t, t+\Delta)$. ∎

Theorem 2. *If the response time of T_i's jobs is at most Θ_i, then at least*

$$\mathcal{B}'(\Delta) = \sup_{0 \leq y \leq \Delta} (Z(y)) \tag{8}$$

time units are available over any interval of length $\Delta \geq 0$, where $Z(y) = \max\left(0,\ \mathcal{B}(y) - \sum_{T_i \in \tau} \min(y, \gamma_i^u(\alpha_i^u(y + \Theta_i)))\right)$. Additionally, (4) for $\mathcal{B}'(\Delta)$ holds with $\widehat{U}' = \widehat{U} - U_{sum}$ and $\sigma'_{tot} = (\widehat{U} \cdot \sigma_{tot} + \sum_{T_i \in \tau}(u_i \cdot \Theta_i + \overline{e_i} \cdot B_i + v_i))/\widehat{U}'$.

Proof: In this paper, we prove (8); derivations of the coefficients $\widehat{U}'$ and σ_{tot} can be found in [11]. Consider an interval $[t, t+y)$, where $y \leq \Delta$. By Defs. 5 and 8, the supply that is available after scheduling the tasks in τ in this interval is

$$\begin{aligned}
&\mathsf{Supply}(t, y) - \sum_{T_i \in \tau} \mathsf{A}(T_i, [t, t+y)) \\
&\quad \{\text{by Def. 6}\} \\
&\geq \max\left(0, \mathcal{B}(y) - \sum_{T_i \in \tau} \mathsf{A}(T_i, [t, t+y))\right) \\
&\quad \{\text{by Lemma 1}\} \\
&\geq \max\left(0, \mathcal{B}(y) - \sum_{T_i \in \tau} \min(y, \gamma_i^u(\alpha_i^u(y + \Theta_i)))\right).
\end{aligned}$$

Additionally, $\mathsf{Supply}(t, \Delta) \geq \sup_{0 \leq y \leq \Delta}(\mathsf{Supply}(t, y))$. ∎

V. Multiprocessor Schedulability Test

In this section, we present the core analysis of our framework in the form of a schedulability test (given in Corollary 1 later in this section) that checks whether a predefined response-time bound Θ_i is not violated for a task T_i.

As noted earlier, the way jobs are prioritized according to Def. 7 is similar to GEDF. A number of GEDF schedulability tests have been developed assuming that jobs arrive periodically or sporadically (e.g., [1], [2], [9]). In this paper, we extend techniques from [1] and [9] in order to incorporate more general job arrivals and execution models.

Similarly to [7], we derive our test by ordering jobs by their priorities and assuming that $T_{\ell,q}$ is the first job for which $f_{\ell,q} > r_{\ell,q} + \Theta_\ell$ holds. We further assume that, for each job $T_{a,b}$ such that $T_{a,b} \prec T_{\ell,q}$,

$$f_{a,b} \leq r_{a,b} + \Theta_a. \tag{9}$$

We consider an interval that includes the time when $T_{\ell,q}$ becomes ready and the latest time when $T_{\ell,q}$ is allowed to complete, which is $r_{\ell,q} + \Theta_\ell$. This interval is computed for each value of $k \in [1, K_\ell]$ (see Def. 3) and δ (defined later in this section), which determine its length, $\delta + \Theta_\ell$. (The range of δ depends on k and ℓ.) During this interval, we consider demand due to competing higher-priority jobs that can interfere with $T_{\ell,q}$. We then perform the following three steps:

S1: Compute the minimum guaranteed supply over the interval of interest, $\mathcal{B}(\delta + \Theta_\ell)$.
S2: Given a finite upper bound $M_\ell^*(\delta, \tau, m)$ on the competing demand and a finite upper bound on the unfinished work due to job $T_{\ell,q}$ and its predecessors, $E_\ell^*(k)$, define a sufficient test for checking whether T_ℓ's response-time bound is not violated by setting $M_\ell^*(\delta, \tau, m) + (m-1) \cdot (E_\ell^*(k) - 1) < \mathcal{B}(\delta + \Theta_\ell)$.
S3: Calculate $M_\ell^*(\delta, \tau, m)$ and $E_\ell^*(k)$ as used in **S2**.

A. Steps S1 and S2

To avoid distracting "boundary cases," we henceforth assume that the schedule being analyzed is prepended with a schedule in which response-time bounds are not violated that is long enough to ensure that all predecessor jobs referenced in the proof exist. Since job priorities remain fixed, we also ignore jobs that have lower priority than $T_{\ell,q}$.

We start the derivation by stating the following lemma and claims. The following lemma specifies the minimum time between the arrivals of jobs $T_{\ell,q-i}$ and $T_{\ell,q}$.

Lemma 2. (Proved in [11]) $r_{\ell,q} - r_{\ell,q-i} \geq \mathcal{A}_\ell^{-1}(i)$.

The next two claims establish a lower bound on the maximum job response time and an upper bound on the finish times of certain jobs that can be used in addition to (9).

Claim 1: $\Theta_\ell \geq \gamma_\ell^u(1)$.

Proof: By (6), $\Theta_\ell \geq \max_{j\geq 1}(\gamma_\ell^u(j) - \mathcal{A}_\ell^{-1}(j-1)) \geq \gamma_\ell^u(1) - \mathcal{A}_\ell^{-1}(0)$. By Def. 3, $\mathcal{A}_\ell^{-1}(0) = 0$. ∎

Claim 2: $f_{\ell,q-K_\ell} \leq r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(K_\ell)$.

Proof: By (9),

$$\begin{aligned} & f_{\ell,q-i} \\ & \quad \leq r_{\ell,q-i} + \Theta_\ell \\ & \quad = r_{\ell,q-i} - r_{\ell,q} + r_{\ell,q} + \Theta_\ell \\ & \qquad \{\text{by Lemma 2}\} \\ & \quad \leq r_{\ell,q} + \Theta_\ell - \mathcal{A}_\ell^{-1}(i). \end{aligned} \tag{10}$$

By (1), $-\mathcal{A}_\ell^{-1}(K_\ell) \leq -\gamma_\ell^u(K_\ell)$. Setting this and $i = K_\ell$ into (10), we get the required result. ∎

Job $T_{\ell,q}$ can violate its response-time bound for the following reasons. If $T_{\ell,q-1}$ completes by time $r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(1)$, then $T_{\ell,q}$ may finish its execution after $r_{\ell,q} + \Theta_\ell$ if, after time $\max(f_{\ell,q-1}, r_{\ell,q})$, higher-priority jobs deprive it of processor time or one or more processors are unavailable.

Alternatively, $T_{\ell,q-1}$ may complete *after* time $r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(1)$, which can happen if the minimum job inter-arrival time for T_ℓ is less than $\gamma_\ell^u(1)$. In this situation, $T_{\ell,q}$ could violate its response-time bound even if it executes uninterruptedly within $[f_{\ell,q-1}, r_{\ell,q} + \Theta_\ell)$. In this case, T_ℓ's response-time bound is violated because $T_{\ell,q-1}$ completes "late," namely after time $r_{\ell,q}$ (recall that, by Claim 1, $\Theta_\ell \geq \gamma_\ell^u(1)$). However, this implies that T_ℓ is pending continuously throughout the interval $[r_{\ell,q-1}, r_{\ell,q} + \Theta_\ell)$, and hence, we can examine the execution of jobs $T_{\ell,q-1}$ and $T_{\ell,q}$ together. In this case, we need to consider the completion time of job $T_{\ell,q-2}$. If $f_{\ell,q-2} \leq r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(2)$, then job $T_{\ell,q}$ may exceed its response-time bound if this job and its predecessor, $T_{\ell,q-1}$, experience interference from higher-priority jobs or some processors are unavailable during the time interval $[\max(f_{\ell,q-2}, r_{\ell,q-1}), r_{\ell,q} + \Theta_\ell)$. On the other hand, if $f_{\ell,q-2} > r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(2)$, then $T_{\ell,q}$ can complete after time $r_{\ell,q} + \Theta_\ell$ even if T_ℓ executes uninterruptedly within $[f_{\ell,q-2}, r_{\ell,q} + \Theta_\ell)$. Continuing by considering predecessor jobs $T_{\ell,q-k}$ in this manner, we will exhaust all possible reasons for the response-time bound violation. Note that it is sufficient to consider only jobs $T_{\ell,q-1}, \ldots, T_{\ell,q-K_\ell}$ since, by Claim 2, $f_{\ell,q-K_\ell} \leq r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(K_\ell)$. Assuming that, for job $T_{\ell,q-k}$, $f_{\ell,q-k} \leq r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(k)$, we define the *problem window* for jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$ as $[r_{\ell,q-k+1}, r_{\ell,q} + \Theta_\ell)$. (This problem window definition is a significant difference when comparing our analysis to prior analysis pertaining to periodic or sporadic systems.)

Definition 9. Let $\lambda \in [1, K_\ell]$ be the smallest integer such that $f_{\ell,q-\lambda} \leq r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(\lambda)$. By Claim 2, such a λ exists.

Claim 3. *T_ℓ is ready (i.e., has a ready job) at each instant of the interval $[r_{\ell,q-k+1}, r_{\ell,q} + \Theta_\ell)$ for each $k \in [1, \lambda]$.*

Proof: To prove the claim, we first show that T_ℓ is ready continuously within $[r_{\ell,q-k+1}, f_{\ell,q})$ for each $k \in [1, \lambda]$. Because T_ℓ is ready within the interval $[r_{\ell,q}, f_{\ell,q})$, this is true for $k = 1$. If $k > 1$ (in which case $\lambda > 1$), then $f_{\ell,q-j} > r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(j)$ for each $j \in [1, \lambda)$, by the selection of λ. From this, we have

$$\begin{aligned} & f_{\ell,q-j} \\ & \quad > r_{\ell,q} + \Theta_\ell - \gamma_\ell^u(j) \\ & \qquad \{\text{because, by (6), } \Theta_\ell \geq \gamma_\ell^u(j) - \mathcal{A}_\ell^{-1}(j-1)\} \\ & \quad \geq r_{\ell,q} - \mathcal{A}_\ell^{-1}(j-1) \\ & \qquad \{\text{by Lemma 2}\} \\ & \quad \geq r_{\ell,q-j+1}. \end{aligned}$$

Thus, the intervals $[r_{\ell,q-j}, f_{\ell,q-j})$ and $[r_{\ell,q-j+1}, f_{\ell,q-j+1})$, where consecutive jobs of T_ℓ are ready, overlap. Therefore, T_ℓ is ready continuously within $[r_{\ell,q-j}, f_{\ell,q})$ for each $j \in [1, \lambda)$, and hence, T_ℓ is ready continuously within $[r_{\ell,q-k+1}, f_{\ell,q})$ for each $k \in [2, \lambda]$. The claim follows from $[r_{\ell,q-k+1}, r_{\ell,q} + \Theta_\ell) \subset [r_{\ell,q-k+1}, f_{\ell,q})$; to see this, note that $f_{\ell,q} > r_{\ell,q} + \Theta_\ell$ holds, since $T_{\ell,q}$ violates its response-time bound. ∎

Because $T_{\ell,q}$ violates its response-time bound, after time $r_{\ell,q-\lambda+1}$, there are other higher-priority jobs that deprive T_ℓ of processor time or one or more processors are unavailable.

Definition 10. Let $W(T_{i,y}, t)$ denote the remaining execution time for job $T_{i,y}$ (if any) after time t. Let $W(T_i, t) = \sum_{T_{i,y} \preceq T_{\ell,q}} W(T_{i,y}, t)$. It can be shown that

$$W(T_\ell, r_{\ell,q-\lambda+1}) \leq r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1}. \tag{11}$$

In Fig. 2, which shows a response-time bound violation for job $T_{\ell,q}$ where $\lambda = 1$, $W(T_\ell, r_{\ell,q-\lambda+1})$ corresponds to the execution demand of job $T_{\ell,q}$ and the unfinished work of job $T_{\ell,q-1}$ at time $r_{\ell,q}$.

Definition 11. Let $\Gamma_\lambda \subseteq [r_{\ell,q-\lambda+1}, r_{\ell,q} + \Theta_\ell)$ be the set of intervals where no available processor is idle as shown in Fig. 2. Let $\overline{\Gamma_\lambda} = [r_{\ell,q-\lambda+1}, r_{\ell,q} + \Theta_\ell) \setminus \Gamma_\lambda$. We let $|\Gamma_\lambda|$ ($\overline{\Gamma_\lambda}$) denote the total length of the intervals in Γ_λ, ($|\overline{\Gamma_\lambda}|$).

The lemma below is used to establish a lower bound on the competing workload within the interval $[r_{\ell,q-\lambda+1}, r_{\ell,q} + \Theta_\ell)$.

Lemma 3. *If the response-time bound for $T_{\ell,q}$ is violated (as we have assumed), then $|\Gamma_\lambda| = r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1} - W(T_\ell, r_{\ell,q-\lambda+1}) + 1 + \mu$, where $\mu \geq 0$. (Note that, by (11), $|\Gamma_\lambda| > 0$.) Additionally, T_ℓ executes within each instant of $\overline{\Gamma_\lambda}$, and $|\overline{\Gamma_\lambda}| = W(T_\ell, r_{\ell,q-\lambda+1}) - 1 - \mu$.*

Proof: Suppose, contrary to the statement of the lemma, that the response-time bound for $T_{\ell,q}$ is violated and

$$|\Gamma_\lambda| < r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1} - W(T_\ell, r_{\ell,q-\lambda+1}) + 1. \tag{12}$$

Under these conditions, the total length of the intervals in $\overline{\Gamma_\lambda}$, where at least one available processor is idle, is $r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1} - |\Gamma_\lambda| \overset{\{\text{by (12)}\}}{>} W(T_\ell, r_{\ell,q-\lambda+1}) - 1$. Thus, this total length is at least $W(T_\ell, r_{\ell,q-\lambda+1})$, as time is integral. By Claim 3, T_ℓ executes at each time $t \in \overline{\Gamma_\lambda}$, and thus completes by time $r_{\ell,q} + \Theta_\ell$, which is a contradiction. $|\overline{\Gamma_\lambda}|$ can be found as $|\overline{\Gamma_\lambda}| = r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1} - |\Gamma_\lambda| = W(T_\ell, r_{\ell,q-\lambda+1}) - 1 - \mu$. ■

The next few definitions are used to set up an extension of the problem window to the left so that a greater portion of the workload can be considered. This technique is adapted from [1], [9] and improves the accuracy of the test.

Definition 12. Let $\tau_p(t) = \{T_h \mid$ for some y, $T_{h,y}$ is ready at time t and $T_{h,y} \preceq T_{\ell,q}\}$. (The subscript p denotes the fact that these jobs have higher or equal priority.)

Definition 13. Let $t_0(k) \le r_{\ell,q-k+1}$ be the earliest instant such that $\forall t \in [t_0(k), r_{\ell,q-k+1})$, $|\tau_p(t)| \ge m$ or fewer than $|\tau_p(t)|$ tasks from $\tau_p(t)$ execute at time t. If such an instant does not exist, then let $t_0(k) = r_{\ell,q-k+1}$.

Def. 13 generalizes the well-known concept of an *idle instant* in uniprocessor scheduling as illustrated in Fig. 2. The following claim is used to calculate the competing demand within the interval $[t_0(\lambda), r_{\ell,q-\lambda+1})$.

Claim 4. *No available processor is idle within* $[t_0(\lambda), r_{\ell,q-\lambda+1})$.

Proof: Suppose that an available processor is idle at time $t \in [t_0(\lambda), r_{\ell,q-\lambda+1})$. Because the scheduler being analyzed is work-conserving, all tasks in $\tau_p(t)$ execute at time t and thus $|\tau_p(t)| \le m - 1$, which violates Def. 13. ■

Our schedulability test for task T_ℓ is based upon summing the competing demand of tasks in τ within the interval $[t_0(\lambda), r_{\ell,q} + \Theta_\ell)$, which has length $r_{\ell,q} - t_0(\lambda) + \Theta_\ell$, and the unavailable time within this interval.

Definition 14. Let $E^*_\ell(k)$ be a finite function of k such that $W(T_\ell, r_{\ell,q-\lambda+1}) \le E^*_\ell(\lambda)$. Let $W(t) = \sum_{T_i \in \tau} W(T_i, t)$. Let $M^*_\ell(\delta, \tau, m)$ be a finite function of δ, m, and τ such that $W(t_0(\lambda)) \le M^*_\ell(r_{\ell,q} - t_0(\lambda), \tau, m)$. The function $M^*_\ell(\delta, \tau, m)$ upper-bounds the competing demand due to higher-priority jobs and predecessors of $T_{\ell,q}$ over intervals of length $\delta + \Theta_\ell$. (As mentioned earlier at the beginning of Sec. V, $M^*_\ell(\delta, \tau, m)$ and $E^*_\ell(k)$ are calculated in order to test whether the response-time bound of T_ℓ is not violated. Later, in Sec. V-B, we explain how $M^*_\ell(\delta, \tau, m)$ and $E^*_\ell(k)$ are calculated.)

Definition 15. We require that there exists a constant $H_\ell \ge 0$ such that, for all $\delta \ge 0$,

$$M^*_\ell(\delta, \tau, m) \le U_{sum} \cdot \delta + H_\ell. \tag{13}$$

This requirement is reasonable because the growth rate of the total demand over an interval of interest, which has length $r_{\ell,q} - t_0(\lambda) + \Theta_\ell$, cannot be larger than the total long-term utilization of the tasks in τ for large values of $r_{\ell,q} - t_0(\lambda)$. This also allows us to upper-bound our test's computational complexity. Henceforth, we omit the last four arguments of M^*_ℓ.

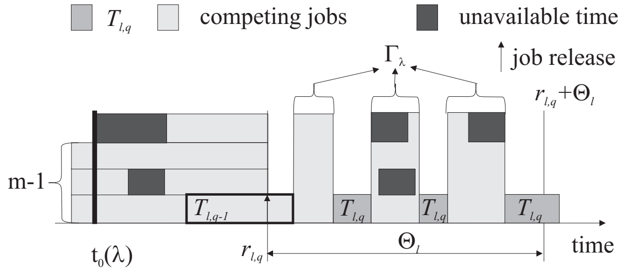


Figure 2. Conditions for response-time bound violation for $\lambda = 1$.

Definition 16. Let $\delta^{\max}_\ell(k) = \big\lfloor (H_\ell + (m-1) \cdot (E^*_\ell(k) - 1) + \widehat{U} \cdot \sigma_{tot} - \Theta_\ell \cdot \widehat{U}) / (\widehat{U} - U_{sum}) \big\rfloor$.

The following theorem will be used to define our schedulability test.

Theorem 3. *If the response-time bound Θ_ℓ is violated for $T_{\ell,q}$ (as we have assumed), then, for some $k \in [1, K_\ell]$ and $\delta \in [\mathcal{A}^{-1}_\ell(k-1), \delta^{\max}_\ell(k)]$,*

$$M^*_\ell(\delta) + (m-1) \cdot (E^*_\ell(k) - 1) \ge \mathcal{B}(\delta + \Theta_\ell). \tag{14}$$

Proof: Consider job $T_{\ell,q}$, $k = \lambda$, and time instants $r_{\ell,q-\lambda+1}$ and $t_0(\lambda)$ as defined in Defs. 9 and 13. To establish (14) (with δ as defined later), we sum the processor allocations within the intervals $[t_0(\lambda), r_{\ell,q-\lambda+1}) \cup \Gamma_\lambda$ and $\overline{\Gamma_\lambda}$. By Def. 11 and Claim 4, the total processor allocation (including unavailable time) within $[t_0(\lambda), r_{\ell,q-\lambda+1}) \cup \Gamma_\lambda$ is $m \cdot (r_{\ell,q-\lambda+1} - t_0(\lambda)) + m \cdot |\Gamma_\lambda|$ (see Fig. 2; note that $r_{\ell,q-\lambda+1} = r_{\ell,q}$ here). Also, Lemma 3 implies that the total processor allocation (including unavailable time) within $\overline{\Gamma_\lambda}$ is at least $W(T_\ell, r_{\ell,q-\lambda+1}) - 1 - \mu$, where $\mu \ge 0$.

The total processor allocation (including unavailable time) within $[t_0(\lambda), r_{\ell,q} + \Theta_\ell)$ is thus at least $m \cdot (r_{\ell,q-\lambda+1} - t_0(\lambda)) + m \cdot |\Gamma_\lambda| + |\overline{\Gamma_\lambda}| \overset{\{\text{by Lemma 3}\}}{=} m \cdot (r_{\ell,q-\lambda+1} - t_0(\lambda)) + m \cdot (r_{\ell,q} + \Theta_\ell - r_{\ell,q-\lambda+1} - W(T_\ell, r_{\ell,q-\lambda+1}) + 1 + \mu) + W(T_\ell, r_{\ell,q-\lambda+1}) - 1 - \mu = m \cdot (r_{\ell,q} + \Theta_\ell - t_0(\lambda)) - (m-1) \cdot (W(T_\ell, r_{\ell,q-\lambda+1}) - 1) + (m-1) \cdot \mu$.

Let $\mathsf{Res}_h([t_0(\lambda), r_{\ell,q} + \Theta_\ell))$ be the amount of time that is not available on processor h at time instants in the interval $[t_0(\lambda), r_{\ell,q} + \Theta_\ell)$. By Defs. 10 and 14, the allocation of jobs within $[t_0(\lambda), r_{\ell,q} + \Theta_\ell)$ is upper-bounded by $W(t_0(\lambda))$ (recall that we are ignoring lower-priority jobs). Thus,

$$\begin{aligned} &W(t_0(\lambda)) + \sum_{h=1}^{m} \mathsf{Res}([t_0(\lambda), r_{\ell,q} + \Theta_\ell)) \\ &\ge m \cdot (r_{\ell,q} - t_0(\lambda) + \Theta_\ell) - (m-1) \cdot (W(T_\ell, r_{\ell,q-\lambda+1}) - 1). \end{aligned} \tag{15}$$

We next calculate an upper bound on $\mathsf{Res}_h([t_0(\lambda), r_{\ell,q}+\Theta_\ell))$. For processor h and the interval $[t_0(\lambda), r_{\ell,q}+\Theta_\ell)$, by Def. 5,

$$\begin{aligned}&\mathsf{Res}_h([t_0(\lambda), r_{\ell,q}+\Theta_\ell))\\ &\quad= (r_{\ell,q}-t_0(\lambda)+\Theta_\ell)-\mathsf{supply}_h(t_0(\lambda), r_{\ell,q}+\Theta_\ell-t_0(\lambda))\end{aligned} \tag{16}$$

Summing (16) for all h, we have

$$\begin{aligned}&\sum_{h=1}^{m}\mathsf{Res}_h([t_0(\lambda), r_{\ell,q}+\Theta_\ell))\\ &\quad= \sum_{h=1}^{m}\big((r_{\ell,q}-t_0(\lambda)+\Theta_\ell)\\ &\qquad\qquad - \mathsf{supply}_h(t_0(\lambda), r_{\ell,q}-t_0(\lambda)+\Theta_\ell)\big)\\ &\qquad\{\text{by Def. 5}\}\\ &\quad= m\cdot(r_{\ell,q}-t_0(\lambda)+\Theta_\ell)-\mathsf{Supply}(t_0(\lambda), r_{\ell,q}-t_0(\lambda)+\Theta_\ell)\\ &\qquad\{\text{by Def. 6}\}\\ &\quad\le m\cdot(r_{\ell,q}-t_0(\lambda)+\Theta_\ell)-\mathcal{B}(r_{\ell,q}-t_0(\lambda)+\Theta_\ell).\end{aligned} \tag{17}$$

Setting (17) into (15), we have

$$\begin{aligned}&W(t_0(\lambda))+m\cdot(r_{\ell,q}-t_0(\lambda)+\Theta_\ell)-\mathcal{B}(r_{\ell,q}-t_0(\lambda)+\Theta_\ell)\\ &\ge m\cdot(r_{\ell,q}-t_0(\lambda)+\Theta_\ell)-(m-1)\cdot(W(T_\ell, r_{\ell,q-\lambda+1})-1).\end{aligned}$$

Rearranging the terms in the above inequality, we have

$$\begin{aligned}&W(t_0(\lambda))+(m-1)\cdot(W(T_\ell, r_{\ell,q-\lambda+1})-1)\\ &\ge \mathcal{B}(r_{\ell,q}-t_0(\lambda)+\Theta_\ell).\end{aligned}$$

Setting $E^*_\ell(\lambda)$ and $M^*_\ell(r_{\ell,q}-t_0(\lambda))$ as defined in Def. 14 into the inequality above, we get

$$\begin{aligned}&M^*_\ell(r_{\ell,q}-t_0(\lambda))+(m-1)\cdot(E^*_\ell(\lambda)-1)\\ &\ge \mathcal{B}(r_{\ell,q}-t_0(\lambda)+\Theta_\ell).\end{aligned}$$

Setting $r_{\ell,q}-t_0(\lambda)=\delta$ in the inequality above we get (14). (Note that, by Def. 9, $\lambda\in[1,K_\ell]$.) By Def. 13 and Lemma 2, $\delta=r_{\ell,q}-t_0(\lambda)\ge r_{\ell,q}-r_{\ell,q-\lambda+1}\ge\mathcal{A}^{-1}_\ell(\lambda-1)$.

Our remaining proof obligation is to establish the stated range for δ. By (13) and (14),

$$U_{sum}\cdot\delta+H_\ell+(m-1)\cdot(E^*_\ell(k)-1)\ge\mathcal{B}(\delta+\Theta_\ell). \tag{18}$$

Applying (4) to (18), we have

$$\begin{aligned}U_{sum}\cdot\delta+H_\ell+(m-1)\cdot(E^*_\ell(k)-1)\\ &\ge \max(0, \widehat{U}\cdot(\delta+\Theta_\ell-\sigma_{tot}))\\ &\ge \widehat{U}\cdot(\delta+\Theta_\ell-\sigma_{tot}).\end{aligned}$$

Solving the latter inequality for δ, we have $\delta\le(H_\ell+(m-1)\cdot(E^*_\ell(k)-1)+\widehat{U}\cdot\sigma_{tot}-\Theta_\ell\cdot\widehat{U})/(\widehat{U}-U_{sum})$. Because δ is integral (as $r_{\ell,q}$ and $t_0(k)$ are integral), by Def. 16, $\delta\le\delta^{\max}_\ell(k)$. The theorem follows. ■

Corollary 1. (Schedulability Test) *If, for task T_ℓ,* (14) *does not hold for each $k\in[1,K_\ell]$ and $\delta\in[\mathcal{A}^{-1}_\ell(k-1), \delta^{\max}_\ell(k)]$, then the response-time bound for T_ℓ is not violated.*

The term $(m-1)\cdot(E^*_\ell(k)-1)$ in (14) can be large if u_ℓ and Θ_ℓ are large. For large values of Θ_ℓ and certain schedulers such as GEDF and FIFO, this term can be replaced with a smaller term proportional to $\max(m-F-1,0)\cdot E^*_\ell(k)$, where F is the number of processors that are always available (see Def. 6). This can be done because, under GEDF and FIFO, the problem job $T_{\ell,q}$ and its predecessors cannot be preempted by other jobs after a certain time point unless the competing demand carried from previous time instants is sufficiently large (see [11] for details).

B. Step **S3** *(Calculating $M^*_\ell(\delta)$ and $E^*_\ell(k)$)*

Note that we did not make any assumptions above about how jobs are scheduled except that the jobs of each task execute sequentially and jobs are prioritized as in Def. 7. Therefore, Corollary 1 is applicable to all fixed job-priority scheduling policies (these policies include preemptive variants of GEDF, FIFO, static-priority policies, and their various combinations) provided the functions $M^*_\ell(\delta)$ (and its linear upper bound in Def. 15) and $E^*_\ell(k)$ are known. $M^*_\ell(\delta)$ and $E^*_\ell(k)$ can be derived for a particular algorithm by extending techniques from previously-published papers on the schedulability of sporadic tasks [1], [9] to incorporate more general arrival and execution patterns.

In the extended version of this paper [11], we derive $E^*_\ell(k)$, $M^*_\ell(\delta)$, and the constant H_ℓ in Def. 15 for a prioritization scheme in which $\chi_{i,j}=r_{i,j}+D_i$, where D_i is a constant (preemptive GEDF and FIFO prioritizations are subcases).

Given an expression for H_ℓ, we can compute $\delta^{\max}_\ell(k)$ in Def. 16 for any given k. Given expressions for $\delta^{\max}_\ell(k)$, $M^*_\ell(\delta)$, and $E^*_\ell(k)$, we can apply Corollary 1 to check that each task $T_\ell\in\tau$ meets its response-time bound. In the next section, we identify conditions under which the test is applicable and discuss its time complexity.

VI. Computational Complexity of the Test

According to Corollary 1, (14) needs to be checked for violation for all $k\in[1,K_\ell]$ and $\delta\in[\mathcal{A}^{-1}_\ell(k-1), \delta^{\max}_\ell(k)]$.

Theorem 4. *The time complexity of the presented test is pseudo-polynomial if there exists a constant c such that $U_{sum}\le c<\widehat{U}$.*

Proof: We start with estimating the complexity of checking (14). The values of $\alpha^u_i(\Delta)$, $\gamma^u_i(k)$, $\mathcal{A}^{-1}_i(k)$, and $\mathcal{B}(\Delta)$ can be computed in constant time if $\alpha^u_i(\Delta)$ and $\gamma^u_i(k)$ consist of periodic and aperiodic piecewise-linear parts and $\mathcal{B}(\Delta)$ is also piecewise-linear. These assumptions are used in prior work on the Real-Time Calculus Toolbox [17] and are sufficient for practical purposes.

It can be shown, that, under these assumptions, $M^*_\ell(\delta)$ for a given value of δ can be computed in $O(n)$ time, where n is the number of tasks [11]. The calculations above need to be repeated for all $k \in [1, K_\ell]$ and all integers in $[\mathcal{A}_\ell^{-1}(k-1), \delta_\ell^{\max}(k)]$. By Def. 16, $\delta_\ell^{\max}(k)$ is finite if its denominator is nonzero. By (5), we have $U_{sum} \leq \widehat{U}$. Therefore, $\delta_\ell^{\max}(k)$ is finite if (5) is strict. ■

Checking that (14) is violated for each integral value in $[\mathcal{A}_\ell^{-1}(k-1), \delta_\ell^{\max}(k)]$ can be computationally expensive. A fixed-point iterative technique can instead be applied so that only a (potentially small) subset of $[\mathcal{A}_\ell^{-1}(k-1), \delta_\ell^{\max}(k)]$ is checked.

VII. Closed-Form Expressions for Response-Time Bounds

In settings where response-time bounds are not known, they need to be calculated. In this section, we present closed-form expressions for the response-time bounds Θ_i under GEDF-like schedulers.

In prior work [7], [10], it has been shown that GEDF (and many other schedulers) ensure a maximum response-time bound of $x+p_i+e_i^{max}$, where $x \geq 0$, for each sporadic task $T_i \in \tau$, if tasks have implicit deadlines, all processors are fully available, and $U_{sum} \leq m$. In this paper, we adopt a similar approach. We seek response-time bounds of the form $\Theta_i = x + \gamma_i^u(K_i) + C_i$, where $x > 0$, K_i is as defined in Def. 3, and $C_h = D_h - \min_{T_i \in \tau}(D_i)$. We let $C_{i,h} = D_i - D_h$.

In the rest of this section, we derive x based upon task parameters and resource availability. The derivation process is similar to finding an upper bound on δ in Theorem 3. In Lemmas 4 and 5 below, we first establish upper bounds on $E_\ell^*(k)$ and $M_\ell^*(\delta)$ as functions of x for the case when the response-time bound is a function of x. We then set the obtained expressions into the schedulability test and solve the resulting inequality for x.

Definition 17. Let $Y_\ell = L_\ell(\max(0, \gamma_\ell^u(K_\ell - 1) - 1) + \gamma_\ell^u(K_\ell) + C_\ell)$, where $L_i(X) = \max(0, u_i \cdot X + R_i \cdot B_i) + v_i$.

Lemma 4. (Proved in [11]): *If $\Theta_\ell = x + \gamma_\ell^u(K_\ell) + C_\ell$, then $E_\ell^*(k) \leq Y_\ell + u_\ell \cdot x$ for $k \in [1, K_\ell]$.*

Definition 18. Let $\mathcal{W}$ be the sum of $m-1$ largest values $u_i \cdot (\gamma_i^u(K_i) + C_i)$.

Lemma 5. (Proved in [11]): *If $\Theta_i = x + \gamma_i^u(K_i) + C_i$ for each task T_i and $\delta \geq 0$, then $M_\ell^*(\delta) \leq U_{sum} \cdot \delta + \sum_{T_i \in \tau} L_i(C_{\ell,i}) + U(m-1) \cdot x + \mathcal{W}$, where $U(m-1)$ is the sum of $m-1$ largest task utilizations.*

Definition 19. Let $a = \min(F+1, m)$, where F is the number of processors that are always available (see Def. 6).

Theorem 5. *If $\widehat{U} - (m-a) \cdot \max(u_i) - U(m-1) > 0$ and $U_{sum} \leq \widehat{U}$, then the maximum response time of any job of T_i is at most $x + \gamma_i^u(K_i) + C_i$, where*

$$x = \max_{T_h \in \tau} \left(\frac{\mathcal{W} + \widehat{U} \cdot \sigma_{tot} + A_h + \sum_{T_i \in \tau} L_i(C_{h,i})}{\widehat{U} - (m-a) \cdot u_h - U(m-1)} \right) + 1 \quad (19)$$

and $A_h = (m-a) \cdot (Y_h - 1) + (a - 1 - \widehat{U}) \cdot (\gamma_h^u(K_h) + C_h) + (a-1) \cdot \max(0, \gamma_h^u(K_h - 1) - 1)$.

Proof: Suppose to the contrary that task T_ℓ violates its response-time bound $\Theta_\ell = x + \gamma_\ell^u(K_\ell) + C_\ell$. Because $x > 0$, $\Theta_\ell > \gamma_\ell^u(k) + C_\ell$ holds for each $k \in [1, K_\ell]$. In [11], we show that under these conditions, for $k = \lambda$ and $\delta \geq \mathcal{A}_\ell^{-1}(\lambda - 1)$, where λ is defined in Def. 9, the following holds.

$$\begin{aligned} &M_\ell^*(\delta) + (m-a) \cdot (E_\ell^*(k) - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(k) + C_\ell + \max(0, \gamma_\ell^u(k-1) - 1)) \\ &\geq \mathcal{B}(\delta + \Theta_\ell) \end{aligned} \quad (20)$$

Setting $k = \lambda$ and the bound for $\mathcal{B}$ given by (4) into (20), we get

$$\begin{aligned} &M_\ell^*(\delta) + (m-a) \cdot (E_\ell^*(\lambda) - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(\lambda) + C_\ell + \max(0, \gamma_\ell^u(\lambda-1) - 1)) \\ &\geq \widehat{U} \cdot (\delta + \Theta_\ell - \sigma_{tot}). \end{aligned}$$

By the selection of Θ_ℓ,

$$\begin{aligned} &M_\ell^*(\delta) + (m-a) \cdot (E_\ell^*(\lambda) - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(\lambda) + C_\ell + \max(0, \gamma_\ell^u(\lambda-1) - 1)) \\ &\geq \widehat{U} \cdot (\delta + x + \gamma_\ell^u(K_\ell) + C_\ell - \sigma_{tot}). \end{aligned}$$

Setting the bounds on $E_\ell^*(\lambda)$ and $M_\ell^*(\delta)$ given by Lemmas 4 and 5 into the inequality above, we have

$$\begin{aligned} &U_{sum} \cdot \delta + \sum_{T_i \in \tau} L_i(C_{\ell,i}) + U(m-1) \cdot x + \mathcal{W} \\ &\quad + (m-a) \cdot (Y_\ell + u_\ell \cdot x - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(\lambda) + C_\ell + \max(0, \gamma_\ell^u(\lambda-1) - 1)) \\ &\quad \geq \widehat{U} \cdot (\delta + x + \gamma_\ell^u(K_\ell) + C_\ell - \sigma_{tot}). \end{aligned}$$

Because $\delta \geq \mathcal{A}_\ell^{-1}(k-1) \geq 0$, where $k \in [1, K_\ell]$, and $U_{sum} \leq \widehat{U}$, by the statement of the theorem,

$$\begin{aligned} &U(m-1) \cdot x + \mathcal{W} + \sum_{T_i \in \tau} L_i(C_{\ell,i}) + (m-a) \cdot (Y_\ell + u_\ell \cdot x - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(\lambda) + C_\ell + \max(0, \gamma_\ell^u(\lambda-1) - 1)) \\ &\quad \geq \widehat{U} \cdot (x + \gamma_\ell^u(K_\ell) + C_\ell - \sigma_{tot}). \end{aligned}$$

After regrouping the terms, we have

$$\begin{aligned} &\mathcal{W} + \sum_{T_i \in \tau} L_i(C_{\ell,i}) + (m-a) \cdot (Y_\ell - 1) \\ &\quad + (a-1) \cdot (\gamma_\ell^u(\lambda) + C_\ell + \max(0, \gamma_\ell^u(\lambda-1) - 1)) \\ &\quad - \widehat{U} \cdot (\gamma_\ell^u(K_\ell) + C_\ell - \sigma_{tot}) \\ &\quad \geq x \cdot (\widehat{U} - (m-a) \cdot u_\ell - U(m-1)). \end{aligned}$$

Solving the above inequality for x, we have

$$x \leq \frac{\mathcal{W} + \widehat{U} \cdot \sigma_{tot} + A_\ell(\lambda) + \sum_{T_i \in \tau} L_i(C_{\ell,i})}{\widehat{U} - (m-a) \cdot u_\ell - U(m-1)}, \qquad (21)$$

where $A_\ell(k) = (m-a) \cdot (Y_\ell - 1) + (a-1) \cdot (\gamma_\ell^u(k) + C_\ell + \max(0, \gamma_\ell^u(k-1) - 1) - \widehat{U} \cdot (\gamma_\ell^u(K_\ell) + C_\ell)$. From Def. 19, we have $m - a \geq 0$ and $a \geq 1$. Thus, since the function $\gamma_\ell^u(k)$ is non-decreasing, $A_\ell(\lambda) \leq A_\ell$, where A_ℓ is defined in the statement of the theorem, and hence, (21) contradicts (19). ■

The result of Theorem 5 is closely related to the results of [7], [10], in which the maximum deadline tardiness of sporadic tasks under different schedulers is studied. In particular, the requirement to have $\widehat{U} - (m-a) \cdot \max(u_i) - U(m-1)$ to be positive is a sufficient condition for maximum job response times (deadline tardiness) to be bounded.

Tightening the bounds. While we cannot assert that the presented expressions are tight, they can be used to constrain an iterative search for tighter response-time bounds, as mentioned earlier at the end of Sec. II. For example, one can calculate initial response-time bounds $\Theta_i^{[0]}$ using Theorem 5 and then try to find tighter bounds of the form $\Theta_i = y \cdot \Theta_i^{[0]}$, where $y < 1$. The multiplier y can be found by performing a binary search in which tentative values of y are tested using Corollary 1. Note that the multiplier y is the same for all tasks. This method worked well in our experimental study (presented in the next section) as all tasks were identical. For non-identical tasks, different multipliers could be used, but we have not yet studied such approaches in any detail. Further work is also needed on the inherent time complexity required to compute response-time bounds.

VIII. Multiprocessor Analysis: A Case Study

Our analysis can be used to derive response-time bounds for workloads that partitioning schemes cannot accommodate and for workloads that cannot be efficiently analyzed under the widely-studied periodic and sporadic models.

To illustrate this, we applied our analysis to a part of a MPEG-2 video decoder application that has been studied previously in [4], [14]. The originally-studied application, shown in Fig. 3(a), is partitioned and mapped onto two PEs, PE1 and PE2. PE1 runs the VLD (variable-length decoding) and IQ (inverse quantization) tasks, while PE2 runs the IDCT (inverse discrete cosine transform) and MC (motion compensation) tasks. The (coded) input bit stream enters this system and is stored in the input buffer B. The macroblocks (frame pieces of size 16×16 pixels) in B are first processed by PE1 and the corresponding partially decoded macroblocks are stored in the buffer B' before being processed by PE2. The resulting stream of fully decoded macroblocks is written into a playout buffer B'' prior to transmission by the output video device. In the above

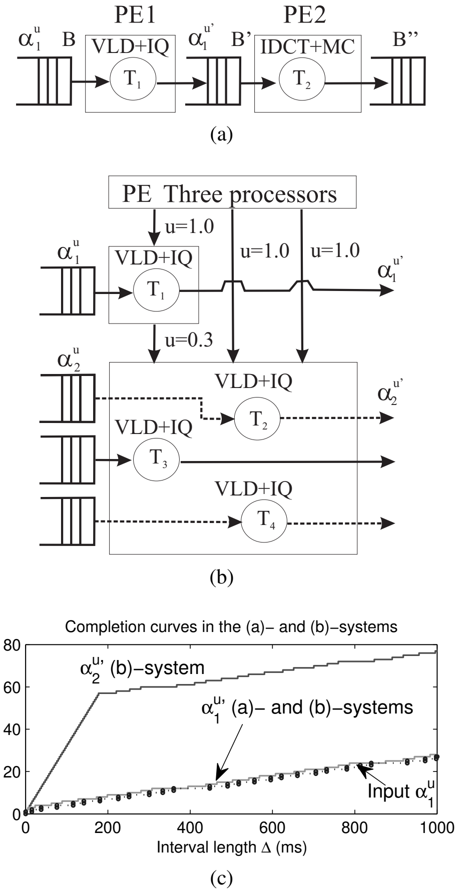


Figure 3. **(a)** A video-processing application. **(b)** Experimental setup. **(c)** Curves $\alpha_1^{u}{}'$ and $\alpha_2^{u}{}'$ and the input α_1^u.

system, the coded input event stream arrives at a constant bit-rate.

Experimental Setup. In our experiments, we considered a variation of the previously-studied system shown in Fig. 3(a) in which PE1 is a three-processor system running four identical VLD+IQ tasks, T_1, T_2, T_3, and T_4. Such a task system could be used in a virtual reality application, where multiple video streams need to be processed. The modified system is illustrated in Fig. 3(b) and explained in further detail below. For conciseness, we refer to the systems in these two insets as the (a)- and (b)-systems, respectively. To assess the usefulness of our analysis, we computed output curves for the four tasks so that they can be used in further analysis. We assumed zero scheduling and system overheads (the inclusion of such overheads in our analysis is beyond the scope of this paper).

The goal of our experiments was to compare different ways of implementing and analyzing the (b)-system. As we shall see, the (b)-system can be implemented on three processors if global scheduling is used; in this case, it can be analyzed using the techniques of this paper but not using prior global schedulability analysis methods. Moreover, if the system is instead partitioned (allowing uniprocessor real-time calculus to be applied on each processor), then four processors are required.

In the analysis, we used a trace of 6×10^5 macroblock processing events obtained in prior work for the VLD+IQ task during a simulation of the (a)-system using a SimpleScalar architecture [4], [14]. We obtained $\gamma_i^u(k)$ as in Def. 1 by examining a repeating pattern of 19,000 consecutive macroblock instruction lengths in the middle of the trace and assuming a 500 MHz processor frequency. We found that all macroblock processing times in the trace are under $\gamma_i^u(1) = 164\mu s$, which we set to be the maximum job execution time (the best-case execution time is $\gamma_i^l(1) = 2\mu s$). The function $\alpha_i^u(\Delta)$ in Def. 2 was obtained by examining macroblock arrival times. We computed $\mathcal{A}_i^{-1}(k)$ in Def. 3 as well as linear bounds for $\alpha_i^u(\Delta)$ and $\gamma_i^u(k)$ as in (2) and (3) using the RTC Toolbox [17].

In the (b)-system, tasks $T_1, \ldots, T_4$, are scheduled on three fully-available processors. Task T_1 is statically prioritized over the other tasks. In such a system, task T_1 can process a time-critical video stream and tasks T_2, T_3, and T_4 can process low-priority video streams. Tasks T_2, T_3, and T_4 are scheduled by GEDF using the supply from two fully-available processors and that remaining on a third processor after accommodating task T_1. In Fig. 3(b), down arrows are used to depict the long-term available utilization on each processor.

Results. To show that existing analysis techniques are inapplicable or are too pessimistic in the given setup, some of the properties of the input streams and the VLD+IQ task need to be emphasized. First, the arrival curve $\alpha_i^u(\Delta)$ is bursty, i.e., several macroblocks can arrive at the same time instant. Second, while $\overline{e_i} = 17.6\mu s$, the maximum execution time of a single macroblock is $164\mu s$, so assuming that each job executes for its worst-case execution time would result in heavy overprovisioning. The long-term per-task utilization is $u_i = R_i \cdot \overline{e_i} = 0.00396 \cdot 17.6 = 0.7$, where $R_i = 0.00396$ is the long-term arrival rate. Finally, the total utilization is $U = \sum_{i=1}^{4} u_i = 2.8$. Therefore, the task set $\{T_1, \ldots, T_4\}$ cannot be partitioned onto three processors (four processors are needed, actually), so *global scheduling is required.*

For the (b)-system, the minimum job inter-arrival time is zero. Moreover, the arriving stream cannot be re-shaped so that the minimum job inter-arrival time is at least $p_i = 25\mu s$ and the long-term arrival rate to be preserved. Because the worst-case job execution time is $e_i^{\max} = \gamma_i^u(1) = 164\mu s$ and the minimum job inter-arrival time is $p_i = 25\mu s$, we have $e_i^{\max}/p_i = 6.59 > 1$. Therefore, the (b)-system *cannot be analyzed using prior results for periodic and sporadic task models*, which require $p_i > 0$ and $e_i^{\max}/p_i \leq 1$.

Fig. 3(c) depicts the job completion curve $\alpha_1^{u\prime}$ (normalized to frames/millisecond assuming 1,584 macroblocks per frame) for task T_1 in the (a)- and (b)-systems, the curve $\alpha_2^{u\prime}$ for task T_2 in the (b)-system, and the input curve α_1^u. (Note that, in the (b)-system, tasks $T_1, \ldots, T_4$ have the same input curve α_1^u, and the completion curves for $T_2, \ldots, T_4$ are the same.) The curves for T_1 in the (a)- and (b)-systems were obtained using prior results in real-time calculus pertaining to uniprocessor systems as implemented in the RTC Toolbox [17]. For the (b)-system, we calculated the maximum response time for T_1 and then applied Theorem 2 to find the supply available to tasks T_2, T_3, and T_4. We then calculated their inital response-time bounds $\Theta_i^{[0]}$ using Theorem 5. Since all three tasks have identical parameters, we calculated tighter bounds by running a binary search as described earlier at the end of Sec. VII. We then computed completion curves using Theorem 1.

The resulting curves have the same long-term completion rate in both systems. Task T_1 has the shortest possible maximum response time in both the (a)- and (b)-systems. However, the large job response times of tasks $T_2, \ldots, T_4$ in the (b)-system cause a larger degree of burstiness in the output event streams. Such burstiness is mainly due to the fact that multiple jobs of the same task arriving at the same time instant can potentially execute for a significant duration of time, causing jobs of non-executing tasks to wait (or be queued). Overall, the (b)-system has the advantage of needing only *three* processors to accommodate four video streams, at the expense of larger buffers for storing partially decoded macroblocks (for approximately 50 frames). With partitioned scheduling, *four* dedicated processors are required.

We conclude this section with a few comments about the running time of the analysis procedures. We have implemented these procedures as a set of MATLAB functions extending the RTC Toolbox. Though the procedure presented in Sec. V has pseudo-polynomial time complexity (like many other schedulability tests presented elsewhere), the time needed to verify response times using Corollary 1 can be large, especially for complex arrival and execution-time patterns. In our experimental study of the (b)-system, we found that the required response-time bounds could be calculated in a couple of minutes (on a 1.7 GHz single-processor desktop system). Again, these bounds were obtained by using Theorem 5, and then refining these bounds using Corollary 1.

IX. Concluding Remarks

In this paper, we have studied a multiprocessor PE, where (partially available) processors are managed by a global scheduling algorithm and jobs are triggered by streams of external events. This work is of importance because it allows workloads to be analyzed for which existing schedulability analysis methods are completely inapplicable (e.g., the system cannot be described efficiently using conventional periodic/sporadic task models) and for which partitioning techniques are unnecessarily restrictive.

The research in this paper is part of a broader effort, the goal of which is to produce a practical compositional framework, based on real-time calculus, for analyzing multiprocessor real-time systems. Towards this goal, the contributions of this paper are as follows. We designed a pseudo-polynomial-time procedure that can be used to test whether job response times occur within specified bounds. Given these bounds, we computed upper and lower bounds on the number of job completion events over any interval of length Δ and a lower bound on the supply available after scheduling all incoming jobs. These bounds can be used as inputs for other PEs thereby resulting in a compositional analysis framework.

A number of unresolved issues of practical importance remain. First, *efficient* methods are needed for determining response-time bounds when they are not specified — this is probably the most important unresolved issue left by this paper. As a partial solution, we provided closed-form expressions for computing response-time bounds, but we do not know how pessimistic they are. Second, the schedulability test itself could possibly be improved by incorporating information about lower bounds on job arrivals and execution times and upper bounds on supply. Third, real-time interfaces as in [4] need to be derived for the multiprocessor case to achieve full compatibility with uniprocessor real-time calculus. Fourth, the inherent pessimism introduced by applying real-time calculus methods on multiprocessors needs to be assessed.

ACKNOWLEDGMENT

Work supported by AT&T, IBM, Intel, and Sun Corps., NSF grants CNS 0834270, CNS 0834132, and CNS 0615197, and ARO grant W911NF-06-1-0425. We are grateful to Linh Thi Xuan Phan for her help with experimental data.

REFERENCES

[1] S. Baruah, "Techniques for multiprocessor global schedulability analysis," in *Proc. of the 28th IEEE Real-Time Systems Symp.*, 2007, pp. 119–128.

[2] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 4, pp. 553–566, 2009.

[3] S. Chakraborty, S. Kunzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. of Design, Automation and Test in Europe - Volume 1*, 2003, p. 10190.

[4] S. Chakraborty, Y. Liu, N. Stoimenov, L. Thiele, and E. Wandeler, "Interface-based rate analysis of embedded systems," in *Proc. of the 27th IEEE Real-Time Systems Symp.*, 2006, pp. 25–34.

[5] R. Cruz, "A calculus for network delay, Part I: Network elements in isolation," *IEEE Tran. on Information Theory*, vol. 37, no. 1, 1991.

[6] ——, "A calculus for network delay, Part II: Network analysis," *IEEE Tran. on Information Theory*, vol. 37, no. 1, 1991.

[7] U. Devi, "Soft real-time scheduling on multiprocessors," Ph.D. dissertation, University of North Carolina, Chapel Hill, NC, 2006.

[8] H. Leontyev and J. Anderson, "Tardiness bounds for FIFO scheduling on multiprocessors," in *Proc. of the 19th Euromicro Conf. on Real-Time Systems*, 2007, pp. 71–80.

[9] ——, "A unified hard/soft real-time schedulability test for global EDF multiprocessor scheduling," in *Proc. of the 29th IEEE Real-Time Systems Symp.*, 2008, pp. 375–384.

[10] ——, "Generalized tardiness bounds for global multiprocessor scheduling," *Real-Time Systems*, 2009, to appear.

[11] H. Leontyev, S. Chakraborty, and J. H. Anderson, "Multiprocessor extensions to real-time calculus (full version)," May 2009. Available: http://cs.unc.edu/~anderson/papers/rtss-09.pdf

[12] A. Maxiaguine, "Modeling multimedia workloads for embedded system design," Ph.D. dissertation, ETH Zurich, 2005.

[13] A. K. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Proc. of 7th Real-Time Technology and Applications Symp.*, 2001, pp. 75–84.

[14] L. Phan, S. Chakraborty, and P. Thiagarajan, "A multi-mode real-time calculus," in *Proc. of the 29th IEEE Real-Time Systems Symp.*, December 2008, pp. 59–69.

[15] K. Richter, M. Jersak, and R. Ernst, "A formal approach to MpSoC performance verification," *IEEE Computer*, vol. 36, no. 4, pp. 60–67, 2003.

[16] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *Proc. of the 20th Euromicro Conf. on Real-Time Systems*, 2008, pp. 181–190.

[17] E. Wandeler and L. Thiele, "Real-Time Calculus (RTC) Toolbox," 2006. [Online]. Available: http://www.mpa.ethz.ch/Rtctoolbox

[18] J. Wu, J.-C. Liu, and W. Zhao, "On schedulability bounds of static priority schedulers," in *Proc. of the 11th IEEE Real Time and Embedded Technology and Applications Symposium*, 2005, pp. 529–540.

[19] F. Zhang and A. Burns, "Schedulability Analysis for Real-Time Systems with EDF Scheduling," University of York, Department of Computer Science, Tech. Rep. YCS-2008-426, 2008.