Multiprocessor Extensions to Real-Time Calculus

Hennadiy Leontyev[1] Samarjit Chakraborty[2] James H. Anderson[1]
[1]Department of Computer Science, University of North Carolina at Chapel Hill
[2]Department of Computer Science, National University of Singapore
E-mail: {leontyev, anderson}@cs.unc.edu, samarjit@comp.nus.edu.sg

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. While a number of such techniques have been recently proposed in the literature, none of them handle multiprocessor processing elements. On the other hand, within the real-time systems community, a number of scheduling algorithms and analysis techniques have been proposed that directly target multiprocessors. In this paper, we present a compositional analysis framework that is obtained by extending the real-time calculus framework to incorporate multiprocessors in which tasks are scheduled using well-known multiprocessor scheduling techniques. We present the basic theory of the resulting extended framework and show its utility using a case study.

1. Introduction

The increasing complexity and heterogeneity of modern embedded platforms have led to a growing interest in compositional modeling and analysis techniques [9]. 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.

To enable more accurate analysis, a compositional technique was proposed in [9], which relies on well-known schedulability and timing-analysis results. As illustrated in Fig. 1(a), the system architecture considered in [9] is very general and consists of multiple processing elements (PEs) that process streams of input data or events (or equivalently, jobs generated by a task). The processed data/events trigger subsequent tasks implemented on other PEs. Different PEs might be connected by communication buses (which can be considered to be PEs as well) or FIFO buffers. Additionally, different PEs might implement different scheduling/arbitration policies. Further, different PEs might assume different task triggering patterns or event models (e.g., periodic, periodic with jitter, sporadic, etc.). While it is possible to analyze each of the individual PEs in isolation using known results from the real-time systems literature (e.g., schedulability analysis of periodic tasks under fixed-priority or earliest-deadline first (EDF) scheduling), it is not clear how to compose the results for different PEs to estimate the timing properties of the full system.

To enable compositions of PEs with different event models, [9] proposed the use of *event model interfaces* (EMIFs). An EMIF translates an event stream that conforms to one event model so that it conforms to another event model. For example, a periodic stream with period p and jitter j can be translated into a sporadic stream with a minimum event separation of $p - j$. Some EMIFs are lossy (e.g., periodic to sporadic), while others are not. If an EMIF does not exist between a pair of event models (e.g., sporadic to periodic), then [9] suggested the use of buffering (or *event adaptation functions*) to physically change the timing properties of the event stream to match the target event model.

An alternative composition technique — often referred to as *real-time calculus* — was proposed in [3] and then subsequently extended in a number of papers (e.g., see [4]). Here, the basic idea is to represent the timing properties of event streams 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 PE within any time interval of length Δ (or the maximum/minimum number of possible task activations within any Δ). The service offered by a PE 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 PE, 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.,

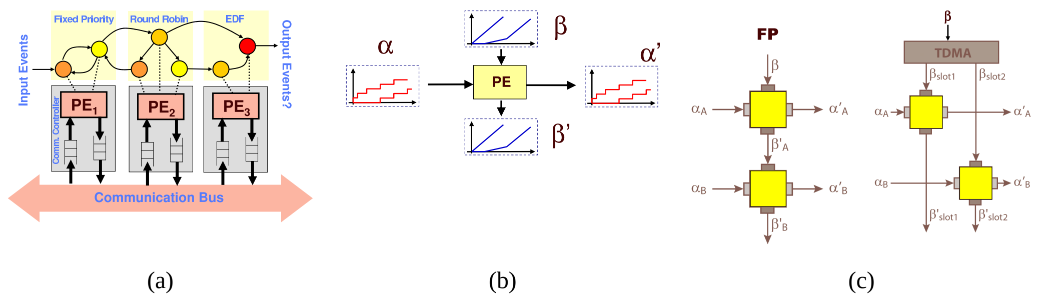

Figure 1. **(a)** System architecture with multiple PEs implementing different scheduling policies. **(b)** Computing the timing properties of the processed stream. **(c)** Scheduling networks.

functions $\alpha^{u\prime}$, $\alpha^{l\prime}$, $\beta^{u\prime}$, and $\beta^{l\prime}$, as illustrated in Fig. 1(b), as well as the maximum backlog and delay experienced by the stream. The computed functions $\alpha^{u\prime}$ and $\alpha^{l\prime}$ can now serve as input to the next PE on which this stream is processed (see Fig. 1(a)). By repeating this procedure until all PEs 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.

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

Our contribution. None of the compositional techniques described above can be used to analyze multiprocessor PEs where *the constituent processors are managed according to a global multiprocessor scheduling algorithm*. There are two reasons why existing compositional techniques need to be extended to incorporate such configurations. 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.)

The main contribution of this paper is an extension of the real-time calculus framework [3, 4] that incorporates symmetric globally-scheduled multiprocessors. In particular, we present a pseudo-polynomial procedure that can be used to test whether event delays on such a multiprocessor reside within specified bounds. Using these bounds, the arrival curves for processed streams can be computed; these curves can in turn be used as input for other PEs thereby resulting in a compositional technique.

The presented test is based upon multiprocessor schedulability tests by Baruah [1] and Leontyev and Anderson [6]. In some aspects, the presented analysis is also similar to results by Bertogna et al. [2], Shin et al. [10], and Zhang and Burns [12]. The main difference between our work and these prior efforts is that we consider more general task arrival and execution models. Also, we consider the case when one or more processors can be partially available, which is similar to [10], where partial availability appears in the context of hierarchical scheduling.

The rest of the paper is organized as follows. Sec. 2 presents our task model. In Sec. 3, the aforementioned test is presented. In Sec. 4, the test's time complexity is discussed. Sec. 5 presents a case study for our analysis. Sec. 6 concludes by summarizing our contributions and listing some directions for future work.

2. Task Model

In this paper, we consider a task set $\tau = \{T_1, T_2, \ldots, T_n\}$. 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*.

Definition 1. $E_i(k)$ denotes an upper bound on the total execution time of any k consecutive jobs of T_i. (We assume $E_i(k) = 0$ for all $k \leq 0$ and $E_i(k) \leq E_i(k+1)$.)

Example 1. If worst-case job execution times follow a repeating pattern as in the multiframe task model [7], then the function $E_i(k)$ can be derived from this pattern, e.g., using the RTC Toolbox [11]. Suppose that task T_i's worst-case job execution times follow a pattern $1, 5, 2, 1, 5, 2, \ldots$. Then, $E_i(1) = 5$, $E_i(2) = 7$, $E_i(3) = 8$, $E_i(4) = 13$, 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 $\Delta > 0$. (We assume $\alpha_i^u(\Delta) = 0$ for all $\Delta \leq 0$.) $\alpha_i(\Delta)$ denotes the pair $(\alpha_i^u(\Delta), \alpha_i^l(\Delta))$.

Definition 3. Let $\alpha_i^+(\Delta) = \lim_{\epsilon \to +0} \alpha_i^u(\Delta + \epsilon)$. This function provides an upper bound on the number of jobs released within any interval $[x, x+\Delta]$, where $x \geq 0$ and $\Delta \geq 0$. (We assume $\alpha_i^+(\Delta) = 0$ for all $\Delta < 0$.)

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 $E_i(k) = k \cdot e_i^{\max}$. The next example illustrates the difference between the functions α_i^u and α_i^+.

Example 3. Consider a task T_i, whose jobs arrive periodically with period p_i. The maximum number of jobs that can arrive within an interval $(x, x + 2 \cdot p_i]$ is thus $\alpha_i^u(2 \cdot p_i) = \left\lceil \frac{2 \cdot p_i}{p_i} \right\rceil = 2$. However, the maximum number of jobs that can arrive within the interval $[x, x+2\cdot p_i]$, is $\alpha_i^+(2 \cdot p_i) = \lim_{\epsilon \to +0} \alpha_i^u(2 \cdot p_i + \epsilon) = 3$. In general, under the sporadic task model, $\alpha_i^+(\Delta) = \left\lfloor \frac{\Delta}{p_i} \right\rfloor + 1$.

Definition 4. Let $\mathcal{A}_i^{-1}(k) = \inf\{\Delta | \alpha_i^u(\Delta) > k\}$. 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 require that there exists $K_i \geq 1$ such that

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

Example 4. (1) is needed in order to prevent task T_i from overloading the system. Under the periodic and sporadic models, $\mathcal{A}_i^{-1}(k) = k \cdot p_i$ and $E_i(k) = k \cdot e_i^{\max}$. If (1) does not hold, then $p_i < e_i^{\max}$, and thus, task T_i's jobs can have unbounded response times.

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

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

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

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

(2) and (3) are needed in order to bound the computation time of the proposed schedulability test. 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. These assumptions are not too restrictive because usually arrival curves are composed of aperiodic and periodic parts for which linear bounds are known (see [11] for an example), and the execution times of consecutive jobs often follow a repeating pattern [7].

Definition 5. Let $u_i = R_i \cdot \overline{e_i}$. This quantity denotes the average long-term utilization of task T_i. We require that $u_i \leq 1$.

Example 5. 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 6. We assume that the available processing capacity is specified using *service functions*. Specifically, the guaranteed time that processor h can provide to the tasks in τ in any time interval of length $\Delta \geq 0$ is within $[\beta_h^l(\Delta), \beta_h^u(\Delta)]$, where

$$\beta_h^l(\Delta) \geq \max(0, \widehat{u_h} \cdot (\Delta - \sigma_h)), \tag{4}$$

for $\widehat{u_h} \in (0, 1]$ and $\sigma_h \geq 0$.

In the above definition, $\widehat{u_h}$ is the total long-term utilization available to the tasks in τ on processor h and σ_h is the maximum length of time when the processor can be unavailable. Note that, if processor h is fully available to the tasks in τ, then $\beta_h^u(\Delta) = \beta_h^l(\Delta) = \Delta$.

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

$$\sum_{T_i \in \tau} u_i \leq \sum_{h=1}^{m} \widehat{u_h} \tag{5}$$

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. A job is *ready* if it is released and 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 value $r_{i,j} + D_i$, where D_i is a constant. If $r_{i,j} + D_i < r_{k,h} + D_k$ or $r_{i,j} + D_i = r_{k,h} + D_k \wedge i < k$ or $r_{i,j} + D_i = r_{k,h} + D_k \wedge i = k \wedge j < h$, then the priority of $T_{i,j}$ is higher than that of $T_{k,h}$, which is denoted $T_{i,j} \prec T_{k,h}$.

If D_i is thought of as the relative deadline of a job, then the above prioritization simply defines a global earliest-deadline first (GEDF) scheduler. On the other hand, if $D_i = 0$ for each task T_i, then the scheduler is global FIFO [5].

The technical contribution of this paper is the following. Given a task set $\tau = \{T_1, \ldots, T_n\}$ and a multiprocessor platform characterized by a collection of service functions $\beta_k^l(\Delta)$, 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

$$\Theta_i \geq E_i(1). \tag{6}$$

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.

Assuming that maximum job response times are known, it is possible to characterize the sequences of job completion events for each task T_i in terms of arrival functions $\alpha_i^{u\prime}$ and $\alpha_i^{l\prime}$, which then can serve as inputs to subsequent PEs, thereby resulting in a compositional technique. (The details for doing this are straightforward and are omitted due to space constraints.)

3. Multiprocessor Schedulability Test

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, 6]). In this paper, we extend techniques from [1] and [6] in order to incorporate more general job arrivals and execution models.

Similarly to [6], we derive our test by ordering jobs by their priorities and assuming that $T_{\ell,q}$ is the first job for which $f_{\ell,q} > r_{\ell,q} + \Theta_\ell$. We further assume that, for each job $T_{a,b}$ such that $T_{a,b} \prec T_{\ell,q}$,

$$f_{a,b} \leq r_{a,b} + \Theta_a. \tag{7}$$

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 $r_{\ell,q} + \Theta_\ell$. 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 a lower bound $LB(\tau, m)$ on this demand that is necessary for $T_{\ell,q}$'s response time to exceed Θ_ℓ; **S2:** given a finite upper bound $UB(\tau, m)$ on this competing demand, define a sufficient test for checking whether a task's response-time bound is not violated by setting $UB(\tau, m) < LB(\tau, m)$; **S3:** estimate $UB(\tau, m)$ as used in **S2**.

3.1. Steps S1 and S2

We start the derivation by proving the following claims. Claim 1 below follows from Def. 4.

Claim 1: $r_{\ell,q} - r_{\ell,q-i} \geq \mathcal{A}_\ell^{-1}(i)$.

Claim 2. *For* $i \geq 1$. $f_{\ell,q-i} \leq r_{\ell,q} + \Theta_\ell - A_\ell^{-1}(i)$.

Proof. By (7), $f_{\ell,q-i} \leq r_{\ell,q-i} + \Theta_\ell = r_{\ell,q-i} - r_{\ell,q} + r_{\ell,q} + \Theta_\ell \overset{\{\text{by Claim 1}\}}{\leq} r_{\ell,q} + \Theta_\ell - A_\ell^{-1}(i)$. ☐

Claim 3: $f_{\ell,q-K_\ell} \leq r_{\ell,q} + \Theta_\ell - E_\ell(K_\ell)$.

Proof. By (1), $\mathcal{A}_\ell^{-1}(K_\ell) \geq E_\ell(K_\ell)$. Thus, $-\mathcal{A}_\ell^{-1}(K_\ell) \leq -E_\ell(K_\ell)$. Setting this and $i = K_\ell$ into Claim 2, 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 $r_{\ell,q} + \Theta_\ell - E_\ell(1)$, 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 $r_{\ell,q} + \Theta_\ell - E_\ell(1)$, which can happen if the minimum job inter-arrival time for T_ℓ is less than $E_\ell(1)$. 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 (6), $\Theta_\ell \geq E_\ell(1)$). 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 $f_{\ell,q-2} \leq r_{\ell,q} + \Theta_\ell - E_\ell(2)$, 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 $f_{\ell,q-2} > r_{\ell,q} + \Theta_\ell - E_\ell(2)$, 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 $T_{\ell,q-i}$ 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 3, $f_{\ell,q-K_\ell} \leq r_{\ell,q} + \Theta_\ell - E_\ell(K_\ell)$. Assuming that, for job $T_{\ell,q-i}$, $f_{\ell,q-i} \leq r_{\ell,q} + \Theta_\ell - E_\ell(i)$, we define the *problem window* for jobs $T_{\ell,q-i+1}, \ldots, T_{\ell,q}$ as $[\max(f_{\ell,q-i}, r_{\ell,q-i+1}), r_{\ell,q} + \Theta_\ell)$.

Definition 8. For $i \geq 1$, we set $t_w(i) = \max(\min(r_{\ell,q} + \Theta_\ell - E_\ell(i), f_{\ell,q-i}), r_{\ell,q-i+1})$. Let $k \geq 1$ be the minimum number such that $f_{\ell,q-k} \leq t_w(k)$. Claim 4 below shows that such a k exists.

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.

Claim 4: $k \leq K_\ell$.

Proof. By Claim 3, $f_{\ell,q-K_\ell} \le r_{\ell,q} + \Theta_\ell - E_\ell(K_\ell)$. Setting this inequality into the expression for $t_w(K_\ell)$ in Def. 8, we have $t_w(K_\ell) = \max(f_{\ell,q-K_\ell}, r_{\ell,q-K_\ell+1})$, which implies $f_{\ell,q-K_\ell} \le t_w(K_\ell)$. □

We call task T_ℓ *ready* at time t if there is a ready job of T_ℓ at time t.

Claim 5. *T_ℓ is ready at each instant of the interval $[t_w(k), r_{\ell,q} + \Theta_\ell)$.*

Proof. Consider a job $T_{\ell,q-j}$, where $j \in [1,k)$. By Def. 8, $f_{\ell,q-j} > t_w(j) = \max(\min(r_{\ell,q} + \Theta_\ell - E_\ell(j), f_{\ell,q-j}), r_{\ell,q-j+1})$. This implies $f_{\ell,q-j} > r_{\ell,q-j+1}$. 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 pending, overlap. Therefore, T_ℓ is pending continuously within $[r_{\ell,q-k+1}, f_{\ell,q})$, where k is defined in Def. 8. Also, if T_ℓ is pending at time t, then there is an unfinished job of T_ℓ at time t, and the earliest released such job is ready. Thus, T_ℓ is ready throughout $[r_{\ell,q-k+1}, f_{\ell,q})$. The claim follows from $[t_w(k), r_{\ell,q}+\Theta_\ell) \subset [r_{\ell,q-k+1}, f_{\ell,q})$ because $t_w(k) \ge r_{\ell,q-k+1}$, by Def. 8, and $f_{\ell,q} > r_{\ell,q} + \Theta_\ell$, since $T_{\ell,q}$ violates its response-time bound. □

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

If jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$ execute for $x_{\ell,q-k+1}, \ldots, x_{\ell,q}$ time units within the interval $[t_w(k), r_{\ell,q}+\Theta_\ell)$, where $x_{\ell,q-i}$ is the actual execution time of $T_{\ell,q-i}$, then $T_{\ell,q}$ cannot violate its response-time bound. If job $T_{\ell,q}$ executes for less than $x_{\ell,q}$ time units within $[t_w(k), r_{\ell,q} + \Theta_\ell)$, then it executes for at most $x_{\ell,q} - 1$ time units within this interval, as time is integral. Thus, the total time for which jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$, *do not* execute in $[t_w(k), r_{\ell,q} + \Theta_\ell)$ while being ready is at least $r_{\ell,q} + \Theta_\ell - t_w(k) - (\sum_{i=1}^{k}(x_{\ell,q}) - 1) \ge r_{\ell,q} + \Theta_\ell - t_w(k) - (E_\ell(k) - 1) = r_{\ell,q} + \Theta_\ell - t_w(k) - E_\ell(k) + 1$.

Definition 9. Let Γ_k be a subset of the set of intervals within $[t_w(k), r_{\ell,q} + \Theta_\ell)$, where task T_ℓ does not execute while being ready, such that the cumulative length of Γ_k is *exactly* $r_{\ell,q} + \Theta_\ell - t_w(k) - E_\ell(k) + 1$. The total length of Γ_k is denoted $|\Gamma_k| = r_{\ell,q} + \Theta_\ell - t_w(k) - E_\ell(k) + 1$.

Definition 10. We let $\tau_p(t) = \{T_a \mid \text{for some } y, T_{a,y} \text{ is pending at time } t \text{ and } T_{a,y} \preceq T_{\ell,q}\}$.

Definition 11. Let $t_0(k) \le t_w(k)$ be the earliest instant such that $\forall t \in [t_0(k), t_w(k))$, $|\tau_p(t)| \ge m$ or fewer than $|\tau_p(t)|$ tasks from $\tau_p(t)$ execute at time t. If such an instant does not exist, then let $t_0(k) = t_w(k)$.

Def. 11 generalizes the well-known concept of an *idle instant* in uniprocessor scheduling. We call an interval $[t_1, t_2)$ *busy* if no available processor is idle within it.

Claim 6. *The time interval $[t_0(k), t_w(k))$ is busy.*

Proof. Suppose that an available processor is idle at time $t \in [t_0(k), t_w(k))$. 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. 11. □

Definition 12. Let $\delta_\ell(k) = r_{\ell,q} - t_0(k)$. Let $\delta_\ell^{\min}(k) = \min(\max(E_\ell(k) - \Theta_\ell, \mathcal{A}_\ell^{-1}(k) - \Theta_\ell), \mathcal{A}_\ell^{-1}(k-1))$.

Claim 7: $\delta_\ell(k) \ge \delta_\ell^{\min}(k)$.

Proof. By Def. 12, $\delta_\ell(k) = r_{\ell,q} - t_0(k) \ge r_{\ell,q} - t_w(k)$. We lower-bound $r_{\ell,q} - t_w(k)$ as follows. By Def. 8,

$$\begin{aligned}
& r_{\ell,q} - t_w(k) \\
& = r_{\ell,q} - \max(\min(r_{\ell,q} + \Theta_\ell - E_\ell(k), f_{\ell,q-k}), r_{\ell,q-k+1}) \\
& = \min(r_{\ell,q} - \min(r_{\ell,q} + \Theta_\ell - E_\ell(k), f_{\ell,q-k}), \\
& \qquad r_{\ell,q} - r_{\ell,q-k+1}) \\
& = \min(\max(r_{\ell,q} - r_{\ell,q} - \Theta_\ell + E_\ell(k), r_{\ell,q} - f_{\ell,q-k}), \\
& \qquad r_{\ell,q} - r_{\ell,q-k+1}) \\
& = \min(\max(E_\ell(k) - \Theta_\ell, r_{\ell,q} - f_{\ell,q-k}), r_{\ell,q} - r_{\ell,q-k+1}) \\
& \quad \{\text{by (7)}\} \\
& \ge \min(\max(E_\ell(k) - \Theta_\ell, r_{\ell,q} - r_{\ell,q-k} - \Theta_\ell), \\
& \qquad r_{\ell,q} - r_{\ell,q-k+1}) \\
& \quad \{\text{by Claim 1}\} \\
& \ge \min(\max(E_\ell(k) - \Theta_\ell, \mathcal{A}_\ell^{-1}(k) - \Theta_\ell), \mathcal{A}_\ell^{-1}(k-1)) \\
& = \delta_\ell^{\min}(k).
\end{aligned}$$
□

Definition 13. Let $I(T_i)$ be the total amount of time for which jobs of task T_i execute within $[t_0(k), t_w(k)) \cup \Gamma_k$.

Definition 14. Let $M^*(\delta_\ell(k), \ell, k, m, \tau)$ be a finite function of $\delta_\ell(k)$, ℓ, k, m, and τ such that $\sum_{T_i \in \tau} I(T_i) \le M^*(\delta_\ell(k), \ell, k, m, \tau)$. (We will derive an expression for $M^*(\delta_\ell(k), \ell, k, m, \tau)$ later in Sec. 3.2.)

Definition 15. We require that there exist the constants $A \ge 0$ and $H_\ell \ge 0$ such that, for all $\delta_\ell(k) \ge \delta_\ell^{\min}(k)$,

$$M^*(\delta_\ell(k), \ell, k, m, \tau) \le A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell. \quad (8)$$

Henceforth, we omit the last four arguments of M^*.

Definition 16. Let $\delta_\ell^{\max}(k) = \lfloor (H_\ell + A \cdot |\delta_\ell^{\min}(k)| + m \cdot (E_\ell(k) - 1) + \sum_{h=1}^{m} \widehat{u_h} \cdot \sigma_h - \Theta_\ell \cdot \sum_{h=1}^{m} \widehat{u_h}) / (\sum_{h=1}^{m} \widehat{u_h} - A) \rfloor$.

Theorem 1. *If the response-time bound Θ_ℓ is violated for $T_{\ell,q}$, then, for some $k \in [1, K_\ell]$ and $\delta_\ell(k) \in [\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k), \delta_\ell^{\max}(k))]$,*

$$M^*(\delta_\ell(k)) + m \cdot (E_\ell(k) - 1) \ge \sum_{h=1}^{m} \beta_h^l(\delta_\ell(k) + \Theta_\ell). \quad (9)$$

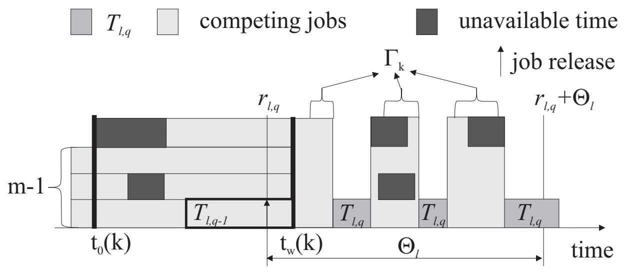

Figure 2. Conditions for response-time bound violation if $k = 1$. (Recall that Γ_k is a *subset* of the intervals where T_ℓ is ready but does not execute.)

Proof. Consider job $T_{\ell,q}$, constant k, and time instants $t_w(k)$ and $t_0(k)$ as defined in Defs. 8 and 11. Let $R_h(\gamma)$ be the amount of time that is not available on processor h at time instants in the set of intervals γ. The amount of available time on processor h during the interval $[t_0(k), r_{\ell,q}+\Theta_\ell)$ is $(r_{\ell,q} + \Theta_\ell - t_0(k)) - R_h([t_0(k), r_{\ell,q} + \Theta_\ell))$, which, by Def. 6, is at least $\beta_h^l(r_{\ell,q} + \Theta_\ell - t_0(k))$. By Def. 12, we thus have

$$(r_{\ell,q} + \Theta_\ell - t_0(k)) - R_h([t_0(k), r_{\ell,q} + \Theta_\ell) \geq \beta_h^l(\delta_\ell(k) + \Theta_\ell). \tag{10}$$

Since $T_{\ell,q}$ does not execute at time instants in Γ_k (refer to Def. 9), each processor at these time instants is either unavailable or executes a task different from T_ℓ as shown in Fig. 2. Also, by Claim 6, no available processor is idle during $[t_0(k), t_w(k))$. By Def. 13, we thus have

$$\begin{aligned}
&\sum_{i=1}^{n} I(T_i) + \sum_{h=1}^{m} R_h([t_0(k), t_w(k)) \cup \Gamma_k) \\
&= m \cdot (|\Gamma_k| + t_w(k) - t_0(k)) \\
&\quad \{\text{by Def. 9}\} \\
&= m \cdot (r_{\ell,q} + \Theta_\ell - t_w(k) - E_\ell(k) + 1 + t_w(k) - t_0(k)) \\
&= m \cdot (r_{\ell,q} + \Theta_\ell - t_0(k) - E_\ell(k) + 1).
\end{aligned}$$

Rearranging the terms in the above inequality, we have

$$\begin{aligned}
&\sum_{i=1}^{n} I(T_i) + m \cdot (E_\ell(k) - 1) \\
&= m \cdot (r_{\ell,q} + \Theta_\ell - t_0(k)) - \sum_{h=1}^{m} R_h([t_0(k), t_w(k)) \cup \Gamma_k) \\
&\quad \{\text{because } [t_0(k), t_w(k)) \cup \Gamma_k \subseteq [t_0(k), r_{\ell,q} + \Theta_\ell)\} \\
&\geq m \cdot (r_{\ell,q} + \Theta_\ell - t_0(k)) - \sum_{h=1}^{m} R_h([t_0(k), r_{\ell,q} + \Theta_\ell)) \\
&= \sum_{h=1}^{m} ((r_{\ell,q} + \Theta_\ell - t_0(k)) - R_h([t_0(k), r_{\ell,q} + \Theta_\ell))).
\end{aligned} \tag{11}$$

Setting (10) into the right-hand side of (11), we have

$$\sum_{i=1}^{n} I(T_i) + m \cdot (E_\ell(k) - 1) \geq \sum_{h=1}^{m} \beta_h^l(\delta_\ell(k) + \Theta_\ell). \tag{12}$$

By Def. 14, $\sum_{i=1}^{n} I(T_i) \leq M^*(\delta_\ell(k))$. Setting this into (12), we have (9). Our remaining proof obligation is to establish the stated range for $\delta_\ell(k)$. By (8) and (9),

$$A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell + m \cdot (E_\ell(k) - 1) \geq \sum_{h=1}^{m} \beta_h^l(\delta_\ell(k) + \Theta_\ell). \tag{13}$$

Applying (4) to (13), we have

$$A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell + m \cdot (E_\ell(k) - 1) \geq \sum_{h=1}^{m} \widehat{u_h}(\delta_\ell(k) + \Theta_\ell - \sigma_h).$$

Solving the latter inequality for $\delta_\ell(k)$ we have $\delta_\ell(k) \leq (H_\ell + A \cdot |\delta_\ell^{\min}(k)| + m \cdot (E_\ell(k) - 1) + \sum_{h=1}^{m} \widehat{u_h} \cdot \sigma_h - \Theta_\ell \cdot \sum_{h=1}^{m} \widehat{u_h})/(\sum_{h=1}^{m} \widehat{u_h} - A)$. Because $\delta_\ell(k)$ is integral, $\delta_\ell(k) \leq \delta_\ell^{\max}(k)$. By Claim 7, $\delta_\ell(k) \geq \delta_\ell^{\min}(k)$. The theorem follows. □

Corollary 1. (Schedulability Test) *If, for task T_ℓ, (9) does not hold for each $k \in [1, K_\ell]$ and $\delta_\ell(k) \in [\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k),\ \delta_\ell^{\max}(k))]$, then the response-time bound for T_ℓ is not violated.*

We did not make any assumptions above about how jobs are scheduled except that jobs of each task execute sequentially. Therefore, Corollary 1 is applicable to all fixed job-priority scheduling policies provided the function $M^*(\delta_\ell(k))$ and its linear upper bound are known. In the next section, we derive the function $M^*(\delta_\ell(k))$ for the case when jobs are prioritized as in Def. 7.

3.2. Finding $M^*(\delta_\ell(k))$

To derive $M^*(\delta_\ell(k))$, we first identify the jobs that may compete with $T_{\ell,q}$ or its predecessors for processor time.

Lemma 1. *Only jobs $T_{a,b}$ such that $T_{a,b} \preceq T_{\ell,q}$ may execute within $[t_0(k), t_w(k)) \cup \Gamma_k$.*

Proof. Suppose to the contrary that a job $T_{a,b} \succ T_{\ell,q}$ executes at time $t \in [t_0(k), t_w(k)) \cup \Gamma_k$. Because the sets $[t_0(k), t_w(k))$ and Γ_k are disjoint, we consider two cases.
Case 1: $t \in [t_0(k), t_w(k))$**.** Since $T_{a,b}$ executes at time t, by Def. 10, each task in $\tau_p(t)$ also executes at time t, which violates Def. 11.
Case 2: $t \in \Gamma_k$**.** By the condition of the case and Def. 9, at time t, a job $T_{\ell,q-i}$, where $i \geq 0$, is ready but not executing. By Def. 7, $T_{a,b} \succ T_{\ell,q} \succeq T_{\ell,q-i}$, where $i \geq 0$. Because $T_{a,b}$ executes at time t, this is a contradiction. □

Definition 17. Let $T_{a,b}$ be the earliest pending job of T_a at time $t_0(k)$. Using Lemma 1, we separate the tasks that may execute within $[t_0(k), t_w(k)) \cup \Gamma_k$ into two disjoint sets:

$$\begin{aligned}\textbf{HC} &= \{T_a :: (T_{a,b} \text{ exists}) \wedge (r_{a,b} < t_0(k)) \wedge (T_{a,b} \preceq T_{\ell,q})\};\\ \textbf{NC} &= \{T_a :: (T_{a,b} \text{ does not exist}) \vee \\ &\qquad [(r_{a,b} = t_0(k)) \wedge (T_{a,b} \preceq T_{\ell,q})]\}.\end{aligned}$$

Here, **HC** denotes "high-priority carry-in" and **NC** denotes "non-carry-in".

Claim 8. *If $\delta_\ell(k) < 0$, then $T_\ell \in$ **HC**.*

Proof. By Def. 12, $t_0(k) = r_{\ell,q} - \delta_\ell(k)$. Assuming $\delta_\ell(k) < 0$, this implies $t_0(k) > r_{\ell,q}$. Therefore, at time $t_0(k)$, $T_{\ell,q}$ or its earliest unfinished predecessor is pending. By Def. 17, this implies $T_\ell \in$ **HC**. □

Claim 9: $|\textbf{HC}| \leq m - 1$.

Proof. By Defs. 10 and 17, **HC** $\subseteq \tau_p(t_0(k) - 1)$. By Def. 11, all tasks in $\tau_p(t_0(k) - 1)$ execute at $t_0(k) - 1$ and $|\tau_p(t_0(k) - 1)| \leq m - 1$. Thus, $|\textbf{HC}| \leq m - 1$. □

We henceforth use $I_{\textbf{NC}}(T_i, \delta_\ell(k))$ and $I_{\textbf{HC}}(T_i, \delta_\ell(k))$ to denote an upper-bound on $I(T_i)$ for the case when T_i is in **NC** and **HC**, respectively. With this notation, we have

$$\sum_{T_i \in \tau} I(T_i) \leq \sum_{T_i \in \textbf{HC}} I_{\textbf{HC}}(T_i, \delta_\ell(k)) + \sum_{T_i \in \textbf{NC}} I_{\textbf{NC}}(T_i, \delta_\ell(k)). \tag{14}$$

The following two lemmas provide expressions for computing $I_{\textbf{NC}}(T_i, \delta_\ell(k))$ and $I_{\textbf{HC}}(T_i, \delta_\ell(k))$. Their proofs can be found in an appendix.

Lemma 2:

$$\begin{aligned}&I_{\textbf{NC}}(T_i, \delta_\ell(k))\\ &= \begin{cases} \min(E_i(\alpha_i^+(\delta_\ell(k) + D_\ell - D_i)), \\ \qquad \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1) & \textit{if } i \neq \ell, \\ \min(E_i(\alpha_i^+(\delta_\ell(k) + D_\ell - D_i) - k), \\ \qquad \delta_\ell(k) - \delta_\ell^{\min}(k)) & \textit{if } i = \ell \wedge \delta_\ell(k) \geq 0, \\ 0 & \textit{otherwise.}\end{cases}\end{aligned}$$

Lemma 3. *Let* $G_i(S, X) = \min(E_i(S), \max(0, X - \mathcal{A}_\ell^{-1}(S-1)) + E_i(S-1))$.

$$\begin{aligned}&I_{\textbf{HC}}(T_i, \delta_{\ell,k})\\ &= \begin{cases} \min(G_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \quad \delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \qquad \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1) & \textit{if } i \neq \ell, \\ \min(G_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i) - k, \\ \quad \delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \qquad \delta_\ell(k) - \delta_\ell^{\min}(k)) & \textit{otherwise.}\end{cases}\end{aligned}$$

To continue our derivation of $M^*(\delta_\ell(k))$, we set

$$\begin{aligned}&M^*(\delta_\ell(k))\\ &= \max\left(\sum_{T_i \in \textbf{HC}} I_{\textbf{HC}}(T_i, \delta_\ell(k)) + \sum_{T_i \in \textbf{NC}} I_{\textbf{NC}}(T_i, \delta_\ell(k))\right),\end{aligned} \tag{15}$$

where max is taken over each choice of **HC** and **NC** subject to the following constraints.

$$\left.\begin{array}{ll} \textbf{NC} \cup \textbf{HC} \subseteq \tau & \textbf{NC} \cap \textbf{HC} = \emptyset \\ \delta_\ell(k) < 0 \Rightarrow T_\ell \in \textbf{HC} & |\textbf{HC}| \leq m - 1 \end{array}\right\} \tag{16}$$

In (16), the constraint $\delta_\ell(k) < 0 \Rightarrow T_\ell \in$ **HC** follows from Claim 8. The constraint $|\textbf{HC}| \leq m - 1$ follows from Claim 9. It is easy to check that $0 \leq I_{\textbf{NC}}(T_i, \delta_\ell(k))$ and $0 \leq I_{\textbf{HC}}(T_i, \delta_\ell(k))$ for each $\delta_\ell(k) \geq \delta_\ell^{\min}(k)$. Thus, the sets maximizing the value $M^*(\delta_\ell(k))$ can be found by adding at most $m - 1$ tasks with the largest positive value of $I_{\textbf{HC}}(T_i, \delta_\ell(k)) - I_{\textbf{NC}}(T_i, \delta_\ell(k))$ to **HC** and adding the remaining tasks to **NC**.

By (14) and (15), $M^*(\delta_\ell(k))$ upper-bounds $\sum_{T_i \in \tau} I(T_i)$ so it complies with Def. 14. In order to use Corollary 1, we are left to find constants A and H_ℓ such that (8) holds so that $M^*(\delta_\ell(k)$ given by (15) complies with Def. 15.

Definition 18. Let $L_i(X) = \max(0, u_i \cdot X + \overline{e_i} \cdot B_i) + v_i$ for any X.

Lemma 4. (Proved in the appendix) *For all $\delta_\ell(k) \geq \delta_\ell^{\min}(k)$, $M^*(\delta_\ell(k)) \leq A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell$, where $A = \sum_{T_i \in \tau} u_i$, $H_\ell = \sum_{T_i \in \tau} L_i(D_\ell - D_i) + U(m-1) \cdot \max(\Theta_i)$, and $U(y)$ is the sum of $\min(y, |\tau|)$ largest utilizations.*

Using the expressions for A and H_ℓ from Lemma 4, we can compute $\delta_\ell^{\max}(k)$ in Def. 16 for any given k. Finally, using the expressions for $\delta_\ell^{\min}(k)$, $\delta_\ell^{\max}(k)$, and $M^*(\delta_\ell(k))$ as given by Defs. 12 and 16 and Equation (15), 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.

4. Computational Complexity of the Test

According to Corollary 1, (9) needs to be checked for violation for all $k \in [1, K_\ell]$ and a set of integers in $[\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k), \delta_\ell^{\max}(k))]$. We start with estimating the complexity of checking (9).

The values of $\alpha_i^u(\Delta)$, $E_i(k)$, $\mathcal{A}_i^{-1}(k)$, and $\beta_h^l(\Delta)$ can be computed in constant time if $\alpha_i^u(\Delta)$ and $E_i(k)$ consist of periodic and aperiodic piecewise-linear parts and $\beta_h^l(\Delta)$ is also piecewise-linear. These assumptions are used in prior work on the Real-Time Calculus Toolbox [11] and are sufficient for practical purposes.

Under these assumptions, by Lemmas 2 and 3, $I_{\mathbf{HC}}(T_i, \delta_\ell(k))$ and $I_{\mathbf{NC}}(T_i, \delta_\ell(k))$ can be computed in $O(1)$ time for each task T_i. Thus, by (15) and (16), computing $M^*(\delta_\ell(k))$ for a given value of $\delta_\ell(k)$ takes $O(n)$ time, where n is the number of tasks, because at most $m-1$ largest positive values $I_{\mathbf{HC}}(T_i, \delta_\ell(k)) - I_{\mathbf{NC}}(T_i, \delta_\ell(k))$ can be selected in $O(n)$ steps [1].

The calculations above need to be repeated for all $k \in [1, K_\ell]$ and all integers in $[\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k), \delta_\ell^{\max}(k))]$. By Def. 16, $\delta_\ell^{\max}(k)$ is finite if its denominator is nonzero. Because, by Lemma 4, $A = \sum_{T_i \in \tau} u_i$, by (5), we have $A = \sum_{T_i \in \tau} u_i \leq \sum_{h=1}^{m} \widehat{u_h}$. Therefore, $\delta_\ell^{\max}(k)$ is finite if (5) is strict. The time complexity of the presented test is thus pseudopolynomial if there exists a constant c such that $\sum_{T_i \in \tau} u_i \leq c < \sum_{h=1}^{m} \widehat{u_k}$. Checking that (9) is violated for each integral value in $[\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k), \delta_\ell^{\max}(k))]$ can be computationally expensive. A fixed-point iterative technique can instead be applied in order to check (9) for a (potentially small) subset of $[\delta_\ell^{\min}(k), \max(\delta_\ell^{\min}(k), \delta_\ell^{\max}(k))]$.

5. Response-Time Analysis: A Case Study

To illustrate the utility of the analysis just derived, we applied it to a part of a video player application. Fig. 3(a) shows an MPEG-2 decoder application that is partitioned and mapped onto two PEs, PE1 and PE2. PE1 runs the VLD and IQ tasks, while PE2 runs the IDCT and MC tasks. The (coded) input bit stream enters this system and is stored in the input buffer B. The macroblocks 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 system, the coded input event stream arrives at a constant bit-rate. This system has been previously studied extensively in [4, 8].

Experimental Setup. In our experiments, we considered variations 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 as shown in Fig. 3(b,c). We computed an upper bound on the response time for each task, i.e., the maximum delay between the time a macroblock is placed in the buffer and the time it is passed downstream. We assumed zero scheduling overheads and zero memory bus contention.

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 system in Fig. 3(a) using a SimpleScalar architecture [4, 8]. We obtained $E_i(k)$ 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 95% of macroblock execution times in the trace are under $E_i(1) = 48\mu s$, which we set to be the maximum macroblock execution time. The function $\alpha_i^u(\Delta)$ as in Def. 2 was obtained by examining macroblock arrival times. We computed $\mathcal{A}_i^{-1}(k)$ and $K_i = 9,339$ in Def. 4 as well as linear bounds for $\alpha_i^u(\Delta)$ and $E_i(k)$ as in (2) and (3) using the RTC Toolbox [11].

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 $\overline{e_i} = 17\mu s$, the maximum execution time of a single macroblock is $48\mu s$, so assuming that each job executes for its worst-case execution time would result in heavy overprovisioning. Finally, the long-term task utilization is $u_i = R_i \cdot \overline{e_i} = 0.0417 \cdot 17 = 0.709$, and the total utilization is $U = \sum_{i=1}^{4} u_i = 2.84$. Therefore, the task set $\{T_1, \ldots, T_4\}$ cannot be partitioned onto three processors, so global scheduling is required.

The system shown in Fig. 3(c) is obtained from that in Fig. 3(b) by introducing four greedy shaper components (GSCs) that separate consecutive job arrivals by p_i time units and hence make $\alpha_i^u(\Delta) = \lceil \frac{\Delta}{p_i} \rceil$. We set $p_i = 24\mu s$ in order to preserve the long-term job arrival rate $R_i = 0.0417 \leq 1/p_i$. In both setups, we set $D_i = 0$, so the scheduler is global FIFO.

Results. We computed maximum task response times in the systems shown in Fig. 3(b,c) using an iterative procedure. We started with setting $\Theta_i = E_i(\alpha_i^+(0)) = 12,706\mu s$ for each task. We then applied Corollary 1 to each of the tasks. If the response-time bound Θ_i for task T_i could not be guaranteed, we increased Θ_i by $10^4\mu s$. We repeated this procedure until all tasks could be guaranteed their response-time bounds. The bounds computed using Corollary 1 were 363 and $313ms$, for systems in Fig. 3(b) and (c), respectively. For the system in Fig. 3(c), the maximum task response time is comprised of the maximum waiting time in the GSC ($63ms$) plus the maximum delay in the PE itself ($250ms$). For the system in Fig. 3(b), the minimum job inter-arrival time is zero. For the system in Fig. 3(c), because the worst-case job execution time $e_i^{\max} = E_i(1) = 48\mu s$ and the minimum job inter-arrival time $p_i = 24\mu s$, we have $e_i^{\max}/p_i = 2 > 1$. Therefore, the task systems shown in Fig. 3(b,c) cannot be analyzed using prior results, which require $p_i > 0$ and $e_i^{\max}/p_i \leq 1$.

The obtained response time bounds are quite large compared to the maximum response time for task T_i running on an dedicated unit-speed processor, which is $62ms$. We believe that such a discrepancy is mainly due to the fact that multiple jobs of the same task arriving at the same time instant can potentially occupy the processor for a significant duration of time, causing jobs of non-executing tasks to wait (or be queued). In the setup in Fig. 3(c), where job arrivals

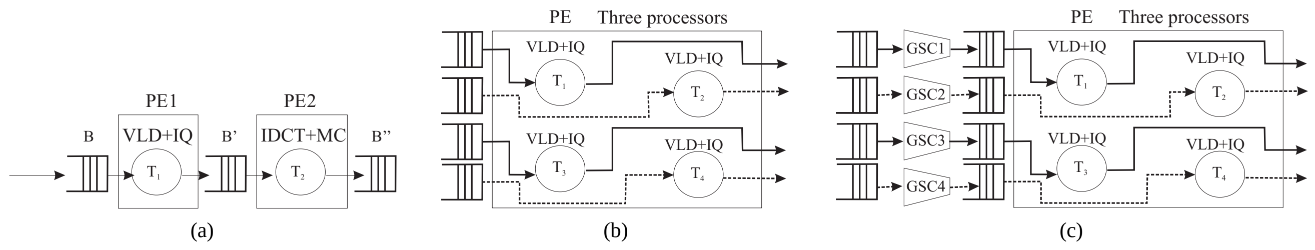

Figure 3. (a) A video-processing application. Experimental setup (b) without and (c) with GSCs.

are separated, the response times are smaller because maintaining the separation between consecutive job arrivals leads to more fair processor allocations among tasks.

The obtained results suggest that the presented 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 widely-studied periodic and sporadic models. However, guaranteed job response times under certain workloads may be large.

6. Conclusion

In this paper, we have presented an extension to the real-time calculus framework. We considered a multiprocessor PE, where (partially available) processors are managed by a global scheduling algorithm and jobs are triggered by streams of external events. We designed a pseudo-polynomial time procedure that can be used to test whether job response times occur within specified bounds. Given these bounds, upper and lower bounds on the number of job completion events over any interval of length Δ can be computed. These bounds can be used as input for other PEs thereby resulting in a compositional technique.

In our analysis, we assumed that response-time bounds are specified. In the future, we plan to extend the analysis in order to explicitly derive upper bounds on job response times from the parameters of the incoming streams, task execution times, and service curves under preemptive and non-preemptive scheduling. Our experimental results suggest that, in the context of stream processing, multiprocessor reservation-based schemes or a prioritization scheme that is cognizant of long-term task progress might be useful. We intend to examine such schemes in future work.

Acknowledgment: Work supported by 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 Symposium*, pages 119–128, December 2007.

[2] M. Bertogna, M. Cirinei, and G. Lipari. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems*, 2008.

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

[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 Symposium*, pages 25–34, December 2006.

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

[6] H. Leontyev and J. Anderson. A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees. In *Proc. of the 20th Euromicro Conf. on Real-Time Systems*, pages 191–200, July 2008.

[7] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23:635–645, 1997.

[8] L. Phan, S. Chakraborty, and P. Thiagarajan. A multi-mode real-time calculus. In *Proc. of the 29th IEEE Real-Time Systems Symposium*, pages 59–69, December 2008.

[9] K. Richter, M. Jersak, and R. Ernst. A formal approach to MpSoC performance verification. *IEEE Computer*, 36(4):60–67, 2003.

[10] 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*, pages 181–190, July 2008.

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

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

Appendix

In this appendix, we prove Lemmas 2, 3, and 4. To prove Lemmas 2 and 3, we first establish some trivial bounds on $I(T_i)$.

Lemma A1: *$I(T_i) \leq \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1$ if $i \neq \ell$, and $I(T_i) \leq \delta_\ell(k) - \delta_\ell^{\min}(k)$ otherwise.*

Proof. If $i \neq \ell$, then, by Def. 13, $I(T_i)$ cannot be larger than the cumulative length of $[t_0(k), t_w(k)) \cup \Gamma_k$. The latter, by Def. 9, is $(t_w(k) - t_0(k)) + |\Gamma_k| = t_w(k) - t_0(k) + r_{\ell,q} + \Theta_\ell - t_w(k) - E_\ell(k) + 1 \overset{\{\text{by Def. 12}\}}{=} \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1$. Alternatively, if $i = \ell$, then $I(T_\ell)$ cannot exceed the length of $[t_0(k), t_w(k))$ because T_ℓ does not execute within Γ_k. We can bound $t_w(k) - t_0(k)$ as follows.

$$\begin{aligned}
& t_w(k) - t_0(k) \\
&\quad = t_w(k) - r_{\ell,q} + r_{\ell,q} - t_0(k) \\
&\qquad \{\text{by Def. 12}\} \\
&\quad = \delta_\ell(k) - (r_{\ell,q} - t_w(k)) \\
&\qquad \{\text{by the proof of Claim 7, } r_{\ell,q} - t_w(k) \geq \delta_\ell^{\min}(k)\} \\
&\quad \leq \delta_\ell(k) - \delta_\ell^{\min}(k).
\end{aligned}$$

□

Lemma A2. *Job $T_{\ell,q-k+1}$ is not ready prior to time $t_w(k)$.*

Proof. Job $T_{\ell,q-k+1}$ is not ready prior to time $\max(f_{\ell,q-k}, r_{\ell,q-k+1})$ since it has to wait until its predecessor finishes. By Def. 8, $f_{\ell,q-k} \leq t_w(k)$. Consider the following two cases.

Case 1: $f_{\ell,q-k} = t_w(k)$**.** By the condition of the case, $\max(f_{\ell,q-k}, r_{\ell,q-k+1}) \geq t_w(k)$. The required result trivially follows.

Case 2: $f_{\ell,q-k} < t_w(k)$**.** Based upon the relationship between $f_{\ell,q-k}$ and $r_{\ell,q} + \Theta_\ell - E_\ell(k)$, we consider two subcases.

Subcase 1: $f_{\ell,q-k} \leq r_{\ell,q} + \Theta_\ell - E_\ell(k)$**.** By Def. 8 and the condition of Subcase 1, we have $t_w(k) = \max(f_{\ell,q-k}, r_{\ell,q-k+1})$, which, by the condition of Case 2, implies $f_{\ell,q-k} < r_{\ell,q-k+1} = t_w(k)$. Therefore, by the condition of Case 2, $\max(f_{\ell,q-k}, r_{\ell,q-k+1}) = t_w(k)$.

Subcase 2: $f_{\ell,q-k} > r_{\ell,q} + \Theta_\ell - E_\ell(k)$**.** By Def. 8, we have $t_w(k) = \max(r_{\ell,q} + \Theta_\ell - E_\ell(k), r_{\ell,q-k+1})$. By the condition of Case 2, this implies $f_{\ell,q-k} < \max(r_{\ell,q} + \Theta_\ell - E_\ell(k), r_{\ell,q-k+1})$. By the condition of Subcase 2, from the latter inequality, we have $f_{\ell,q-k} < r_{\ell,q-k+1}$ and

$$r_{\ell,q-k+1} > r_{\ell,q} + \Theta_\ell - E_\ell(k). \tag{17}$$

By Def. 8,

$$\begin{aligned}
& t_w(k) \\
&\quad = \max(\min(f_{\ell,q-k}, r_{\ell,q} + \Theta_\ell - E_\ell(k)), r_{\ell,q-k+1}) \\
&\qquad \{\text{by the condition of Subcase 2}\} \\
&\quad = \max(r_{\ell,q} + \Theta_\ell - E_\ell(k), r_{\ell,q-k+1}) \\
&\qquad \{\text{by (17)}\} \\
&\quad = r_{\ell,q-k+1}.
\end{aligned} \tag{18}$$

We thus have $\max(f_{\ell,q-k}, r_{\ell,q-k+1}) \overset{\{\text{by (18)}\}}{=} \max(f_{\ell,q-k}, t_w(k)) \overset{\{\text{by the condition of Case 2}\}}{=} t_w(k)$. The required result follows from the two subcases above. □

Corollary A1. *Jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$ do not execute prior to $t_w(k)$.*

Lemma 2:

$$I_{\mathbf{NC}}(T_i, \delta_\ell(k)) = \begin{cases} \min(E_i(\alpha_i^+(\delta_\ell(k) + D_\ell - D_i)), \\ \qquad \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1) & \textit{if } i \neq \ell, \\ \min(E_i(\alpha_i^+(\delta_\ell(k) + D_\ell - D_i) - k), \\ \qquad \delta_\ell(k) - \delta_\ell^{\min}(k)) & \textit{if } i = \ell \wedge \delta_\ell(k) \geq 0, \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. **Case 1:** $i \neq \ell$**.** Because $T_i \in$ **NC**, all of its jobs released prior to $t_0(k)$ are completed by time $t_0(k)$. Thus, the competing demand due to T_i is upper-bounded by the demand due to T_i's jobs released at or after $t_0(k)$ that have higher priority than $T_{\ell,q}$. For such a job $T_{i,j}$, by Def. 7, $r_{i,j} + D_i \leq r_{\ell,q} + D_\ell$, and hence, $r_{i,j} \leq r_{\ell,q} + D_\ell - D_i$. Therefore, the competing demand due to task T_i, $I(T_i)$, is upper-bounded by the total execution time of T_i's jobs released within $[t_0(k), r_{\ell,q} + D_\ell - D_i]$. From Defs. 1 and 3, we have

$$\begin{aligned}
& I(T_i) \\
&\quad \leq E_i(\alpha_i^+(r_{\ell,q} + D_\ell - D_i - t_0(k))) \\
&\qquad \{\text{by Def. 12}\} \\
&\quad = E_i(\alpha_i^+(\delta_\ell(k) + D_\ell - D_i)).
\end{aligned} \tag{19}$$

The stated expression for the case $i \neq \ell$ therefore follows from Lemma A1.

Case 2: $i = \ell \wedge \delta_\ell(k) \geq 0$**.** Applying the reasoning from Case 1 to T_i, we have

$$I(T_i) \leq E_i^u(\alpha_i^+(\delta_{\ell,k} + D_\ell - D_i) - k). \tag{20}$$

The only difference from (19) is that we exclude jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$ from consideration because, by Corollary A1, jobs $T_{\ell,q-k+1}, \ldots, T_{\ell,q}$ do not execute prior to time $t_w(k)$ and, by Def. 9, task T_ℓ does not execute within Γ_k. By (20) and Lemma A1, the stated expression for this case follows.

Case 3: $i = \ell \wedge \delta_\ell(k) < 0$**.** In this case, by Claim 8, task $T_i = T_\ell$ does not belong to **NC**, and hence, we can set $I_{\mathbf{NC}}(T_i, \delta_\ell(k)) = 0$. □

Claim A1. *If $T_{i,y} \preceq T_{\ell,q}$, then $r_{i,y} \leq r_{\ell,q} + D_\ell - D_i$, for $j \geq 0$.*

Proof. The claim immediately follows from Def. 7. □

Definition A1. Let $T_{i,a}$ be the earliest job of T_i that executes within $[t_0(k), t_w(k)) \cup \Gamma_k)$.

Note that, if $T_{i,a}$ does not exist, then $I(T_i) = 0$. We henceforth assume that $T_{i,a}$ exists.

Claim A2. *If $T_{i,a}$ is defined as in Def. A1, then $f_{i,a} > t_0(k)$ and $r_{i,a} > t_0(k) - \Theta_i$.*

Proof. If $f_{i,a} \leq t_0(k)$, then $T_{i,a}$ does not execute within $[t_0(k), t_w(k)) \cup \Gamma_k$, which violates Def. A1. By (7), $f_{i,a} > t_0(k)$ implies $r_{i,a} + \Theta_i > t_0(k)$. □

Definition A2. Let $\kappa_i = \{T_{i,y} : y \geq a \wedge T_{i,y}$ executes in $[t_0(k), t_w(k)) \cup \Gamma_k\}$.

Claim A3. *If $T_{i,y} \in \kappa_i$, then $r_{i,y} \in [r_{i,a}, r_{\ell,q} + D_\ell - D_i]$.*

Proof. By Lemma 1 and Def. A2, $T_{i,y} \preceq T_{\ell,q}$ holds if $T_{i,y}$ is in κ_i. The claim follows from Claim A1. □

Definition A3. Let $\mathsf{A}(T_{i,y}, \gamma)$ be the allocation of $T_{i,y}$ within the set of intervals γ.

Claim A4:

$$I(T_i) = \sum_{T_{i,y} \in \kappa_i} \mathsf{A}(T_{i,y}, [t_0(k), t_w(k)) \cup \Gamma_k). \qquad (21)$$

Proof. The claim follows immediately from Defs. 13, A1, A2, and A3. □

Claim A5. *Let $G_i(S, X) = \min(E_i(S), \max(0, X - \mathcal{A}_i^{-1}(S-1)) + E_i(S-1))$ be as defined in Lemma 3 below. The function $G_i(S, X)$ is a non-decreasing function of the integral argument S.*

Proof. Suppose that $S \geq 1$ is fixed. We compute $G_i(S+1, X)$.

$$\begin{aligned}
&G_i(S+1, X) \\
&= \min(E_i(S+1), \max(0, X - A_\ell^{-1}(S)) + E_i(S)) \\
&\quad \{\text{because } E_i(S) \text{ is a non-decreasing function}\} \\
&\geq E_i(S) \\
&\geq \min(E_i(S), \max(0, X - A_\ell^{-1}(S-1)) + E_i(S-1)) \\
&= G_i(S, X)
\end{aligned}$$
□

Lemma 3. *Let $G_i(S, X) = \min(E_i(S), \max(0, X - \mathcal{A}_i^{-1}(S-1)) + E_i(S-1))$.*

$$I_{\mathbf{HC}}(T_i, \delta_{\ell,k}) = \begin{cases} \min(G_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \quad \delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \quad\quad \delta_\ell(k) + \Theta_\ell - E_\ell(k) + 1) & \textit{if } i \neq \ell, \\ \min(G_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i) - k, \\ \quad \delta_\ell(k) + D_\ell - D_i + \Theta_i), \\ \quad\quad \delta_\ell(k) - \delta_\ell^{\min}(k)) & \textit{otherwise.} \end{cases}$$

Proof. Consider two cases.

Case 1: $i \neq \ell$**.** Let $T_{i,a}$ be as defined in Def. A1. We first rewrite (21).

$$\begin{aligned}
&I(T_i) \\
&\quad = \mathsf{A}(T_{i,a}, [t_0(k), t_w(k)) \cup \Gamma_k)) \\
&\quad\quad + \sum_{T_{i,y} \in \kappa_i \setminus T_{i,a}} \mathsf{A}(T_{i,y}, [t_0(k), t_w(k)) \cup \Gamma_k))
\end{aligned} \qquad (22)$$

We now bound the individual terms in (22). By Claim A2, $T_{i,a}$ finishes its execution at time $f_{i,a} > t_0(k)$, and hence,

$$\begin{aligned}
&\mathsf{A}(T_{i,a}, [t_0(k), t_w(k)) \cup \Gamma_k) \\
&\quad = \min(e_{i,a}, f_{i,a} - t_0(k)) \\
&\quad\quad \{\text{by (7)}\} \\
&\quad \leq \min(e_{i,a}, r_{i,a} + \Theta_i - t_0(k)),
\end{aligned} \qquad (23)$$

where $e_{i,a}$ is the actual execution time of $T_{i,a}$. By (22) and (23),

$$\begin{aligned}
&I(T_i)\\
&\quad\le \min(e_{i,a}, r_{i,a}+\Theta_i - t_0(k))\\
&\qquad + \sum_{T_{i,y}\in\kappa_i\setminus a} \mathsf{A}(T_{i,y},[t_0(k),t_w(k))\cup\Gamma_k)\\
&\quad\le \min\Bigg(e_{i,a} + \sum_{T_{i,y}\in\kappa_i\setminus T_{i,a}} \mathsf{A}(T_{i,y},[t_0(k),t_w(k))\cup\Gamma_k),\\
&\qquad r_{i,a}+\Theta_i - t_0(k)\\
&\qquad + \sum_{T_{i,y}\in\kappa_i\setminus T_{i,a}} \mathsf{A}(T_{i,y},[t_0(k),t_w(k))\cup\Gamma_k)\Bigg). \qquad (24)
\end{aligned}$$

Let $S_i = |\kappa_i|$. Because, by Def. A3, the processor allocation of job $T_{i,y}$ cannot be greater than its execution time, by Def. 1, we have the following.

$$\begin{aligned}
&e_{i,a} + \sum_{T_{i,y}\in\kappa_i\setminus T_{i,a}} \mathsf{A}(T_{i,y},[t_0(k),t_w(k))\cup\Gamma_k)\\
&\qquad\le E_i(S_i) \qquad (25)
\end{aligned}$$

$$\begin{aligned}
&\sum_{T_{i,y}\in\kappa_i\setminus T_{i,a}} \mathsf{A}(T_{i,y},[t_0(k),t_w(k))\cup\Gamma_k)\\
&\qquad\le E_i(S_i-1) \qquad (26)
\end{aligned}$$

By and (24), (25), and (26), we have

$$I(T_i) \le \min(E_i(S_i), r_{i,a}+\Theta_i - t_0(k) + E_i(S_i-1)). \qquad (27)$$

By Claim A3, all jobs $T_{i,y}$ such that $T_{i,y}\in\kappa_i$ are released within $[r_{i,a}, r_{\ell,q}+D_\ell-D_i]$. If $Y \ge S_i$ jobs of T_i are released within $[r_{i,a}, r_{\ell,q}+D_\ell-D_i]$, then $r_{\ell,q}+D_\ell-D_i-r_{i,a} \ge \mathcal{A}_i^{-1}(Y-1)$ by Claim 1, and hence, $r_{i,a} \le r_{\ell,q}+D_\ell-D_i-\mathcal{A}_i^{-1}(Y-1) \le r_{\ell,q}+D_\ell-D_i-\mathcal{A}_i^{-1}(S_i-1)$. We thus have

$$\begin{aligned}
&r_{i,a}+\Theta_i - t_0(k)\\
&\quad\le \max(0,\\
&\qquad r_{\ell,q}+D_\ell-D_i-\mathcal{A}_i^{-1}(S_i-1)+\Theta_i-t_0(k))\\
&\qquad \{\text{by Def. 12}\}\\
&\quad\le \max(0, \delta_\ell(k)+D_\ell-D_i+\Theta_i-\mathcal{A}_i^{-1}(S_i-1)). \qquad (28)
\end{aligned}$$

By (27) and (28), we have

$$\begin{aligned}
&I(T_i)\\
&\quad\le \min(E_i(S_i),\\
&\qquad \max(0,\delta_\ell(k)+D_\ell-D_i+\Theta_i-\mathcal{A}^{-1}(S_i-1))\\
&\qquad + E_i(S_i-1))\\
&\quad= G_i(S_i, \delta_\ell(k)+D_\ell-D_i+\Theta_i), \qquad (29)
\end{aligned}$$

where $G_i(S,X)$ is defined in the statement of the lemma. By Claim A5, the funcion $G_i(S,X)$ is a non-decreasing function of S. We thus can find an upper bound on $I(T_i)$ by setting an upper bound on S_i into (29).

By Claim A3, $S_i = |\kappa_i|$ is at most the number of jobs released within the interval $[r_{i,a}, r_{\ell,q}+D_\ell-D_i]$, which, by Claim A2, is contained within $(t_0(k)-\Theta_i, r_{\ell,q}+D_\ell-D_i]$. We thus upper-bound S_i using Def. 2.

$$\begin{aligned}
&S_i\\
&\le \alpha_i^u(r_{\ell,q}+D_\ell-D_i-t_0(k)+\Theta_i)\\
&\quad \{\text{by Def. 12}\}\\
&= \alpha_i^u(\delta_\ell(k)+D_\ell-D_i+\Theta_i)
\end{aligned}$$

Setting this upper bound on S_i into (29), we have

$$\begin{aligned}
&I(T_i)\\
&\quad\le G_i(\alpha_i^u(\delta_\ell(k)+D_\ell-D_i+\Theta_i),\\
&\qquad \delta_\ell(k)+D_\ell-D_i+\Theta_i).
\end{aligned}$$

The stated expression for the case $i \ne \ell$ therefore follows from Lemma A1.

Case 2: $i = \ell$**.** Repeating the reasoning from the previous case, we find that

$$\begin{aligned}
&I(T_i)\\
&\quad\le G_i(\alpha_i^u(\delta_\ell(k)+D_\ell-D_i+\Theta_i)-k,\\
&\qquad \delta_\ell(k)+D_\ell-D_i+\Theta_i). \qquad (30)
\end{aligned}$$

The only difference with the previous case is that we exclude jobs $T_{\ell,q-k+1},\ldots,T_{\ell,q}$ from consideration because, by Corollary A1, jobs $T_{\ell,q-k+1},\ldots,T_{\ell,q}$ do not execute prior to $t_w(k)$ and, by Def. 9, T_ℓ does not execute within Γ_k. The claimed bound for this case follows from (30) and Lemma A1. □

The following claims and lemma are used to prove Lemma 4.

Claim A6: *$L_i(X+Y) \le L_i(X) + u_i\cdot(Y+|a|)$ for all X and $Y \ge a$.*

Proof. By Def. 5, $u_i > 0$. We consider two cases.

Case 1: $Y \geq 0$. In this case, by Def. 18, $L_i(X + Y) = \max(0, u_i \cdot (X + Y) + \overline{e_i} \cdot B_i) + v_i \leq \max(0, u_i \cdot X + \overline{e_i} \cdot B_i) + v_i + u_i \cdot Y = L_i(X) + u_i \cdot Y$. Because $|a| \geq 0$, the required result follows.

Case 2: $Y < 0$. In this case, by Def. 18, $L_i(X + Y) = \max(0, u_i \cdot (X + Y) + \overline{e_i} \cdot B_i) + v_i \leq \max(0, u_i \cdot X + \overline{e_i} \cdot B_i) + v_i = L_i(X)$. From the statement of the claim and the condition of Case 2, we have $a \leq Y < 0$, and hence, $Y + |a| \geq 0$. We thus have $L_i(X + Y) \leq L_i(X) \leq L_i(X) + u_i \cdot (Y + |a|)$. □

Claim A7: $E_i(\alpha_i^u(X)) \leq E_i(\alpha_i^+(X)) \leq L_i(X)$.

Proof. By Def. 2, $\alpha_i^u(\Delta)$ is a non-decreasing function of Δ. Therefore, $\alpha_i^u(\Delta) \leq \alpha_i^u(\Delta + \epsilon)$ for any $\epsilon > 0$, which implies $\alpha_i^u(\Delta) \leq \lim_{\epsilon \to +0} \alpha_i^u(\Delta + \epsilon)$. The right-hand side of the latter inequality is $\alpha_i^+(\Delta)$ by Def. 3. Thus, $\alpha_i^u(\Delta) \leq \alpha_i^+(\Delta)$. The first inequality of the claim follows from $E_i(k)$ being a non-decreasing function of k by Def. 1. We now prove the second inequality. Because $\alpha_i^+(X) \geq 0$ by Def. 3, we have $E_i(\alpha_i^+(X)) = E_i(\max(0, \alpha_i^+(X)))$. By (3), we have $E_i(\max(0, \alpha_i^+(X))) \leq \overline{e_i} \cdot (\max(0, \alpha_i^+(X))) + v_i$. By (2), $\overline{e_i} \cdot (\max(0, \alpha_i^+(X))) + v_i \leq \overline{e_i} \cdot (\max(0, R_i \cdot X + B_i)) + v_i = \max(0, \overline{e_i} \cdot R_i \cdot X + \overline{e_i} \cdot B_i) + v_i$. By Def. 5, $\max(0, \overline{e_i} \cdot R_i \cdot X + \overline{e_i} \cdot B_i) + v_i = \max(0, u_i \cdot X + \overline{e_i} \cdot B_i) + v_i$. The latter is $L_i(X)$ by Def. 18. □

Lemma A3: $I_{\textbf{HC}}(T_i, \delta_\ell(k)) \leq L_i(\delta_\ell(k) + D_\ell - D_i) + u_i \cdot \Theta_i$ *and* $I_{\textbf{NC}}(T_i, \delta_\ell(k)) \leq L_i(\delta_\ell(k) + D_\ell - D_i)$.

Proof. We prove the first inequality. The second inequality is proved similarly. By Lemma 3, $I_{\textbf{HC}}(T_i, \delta_\ell(k)) \leq E_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i))$. Note that this inequality holds for both $i = \ell$ and $i \neq \ell$ since $k \geq 1$. By Claim A7, $E_i(\alpha_i^u(\delta_\ell(k) + D_\ell - D_i + \Theta_i)) \leq L_i(\delta_\ell(k) + D_\ell - D_i + \Theta_i)$. Because $\Theta_i \geq 0$, by Claim A6, $L_i(\delta_\ell(k) + D_\ell - D_i + \Theta_i) \leq L_i(\delta_\ell(k) + D_\ell - D_i) + u_i \cdot \Theta_i$. □

Lemma 4. *For all* $\delta_\ell(k) \geq \delta_\ell^{\min}(k)$, $M^*(\delta_\ell(k)) \leq A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell$, *where* $A = \sum_{T_i \in \tau} u_i$, $H_\ell = \sum_{T_i \in \tau} L_i(D_\ell - D_i) + U(m-1) \cdot \max(\Theta_i)$, *and* $U(y)$ *is the sum of* $\min(y, |\tau|)$ *largest utilizations.*

Proof. Suppose that the sets **HC** and **NC** subject to (16) maximize the value of the right-hand side of (15). By (15), we have

$$
\begin{aligned}
&M^*(\delta_\ell(k)) \\
&= \sum_{T_i \in \textbf{HC}} I_{\textbf{HC}}(T_i, \delta_\ell(k)) + \sum_{T_i \in \textbf{NC}} I_{\textbf{NC}}(T_i, \delta_\ell(k)) \\
&\quad \{\text{by Lemma A3}\} \\
&\leq \sum_{T_i \in \textbf{HC}} (L_i(\delta_\ell(k) + D_\ell - D_i) + u_i \cdot \Theta_i) \\
&\quad + \sum_{T_i \in \textbf{NC}} L_i(\delta_\ell(k) + D_\ell - D_i) \\
&\quad \{\text{since } \textbf{HC} \cup \textbf{NC} \subseteq \tau\} \\
&\leq \sum_{T_i \in \tau} L_i(\delta_\ell(k) + D_\ell - D_i) + \sum_{T_i \in \textbf{HC}} u_i \cdot \Theta_i \\
&\quad \{\text{because } |\textbf{HC}| \leq m-1 \text{ by (16) and by the} \\
&\quad \text{definition of } U(y) \text{ in the statement of the lemma}\} \\
&\leq \sum_{T_i \in \tau} [L_i(\delta_\ell(k) + D_\ell - D_i)] + U(m-1) \cdot \max(\Theta_i) \\
&\quad \{\text{by Claim A6 (note that, by Claim 7, } \delta_\ell(k) \geq \delta_\ell^{\min}(k))\} \\
&\leq \sum_{T_i \in \tau} [L_i(D_\ell - D_i) + u_i \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|)] \\
&\quad + U(m-1) \cdot \max(\Theta_i) \\
&\quad \{\text{by the definition of } A \text{ and } H_\ell \\
&\qquad \text{in the statement of the lemma}\} \\
&= A \cdot (\delta_\ell(k) + |\delta_\ell^{\min}(k)|) + H_\ell. \qquad \square
\end{aligned}
$$