Bounding Preemptions under EDF and RM Schedulers
Technical Report: MS-CIS-06-07

Arvind Easwaran, Insik Shin, Insup Lee, Oleg Sokolsky
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, 19104 USA
{arvinde, ishin, lee, sokolsky }@cis.upenn.edu

Abstract

Assurance of timing requirements from a real-time system schedule can be obtained using schedulability analysis. Existing analysis techniques ignore the preemption overhead incurred by the system, when the tasks are scheduled using preemptive schedulers. Preemptive schedulers can preempt the execution of a real-time task when a higher priority task is released by the system. Every job preemption will then induce an execution overhead because of the need to swap task contexts. In this paper we derive bounds on the number of preemptions incurred by a real-time task system scheduled using RM or EDF schedulers. We only consider task systems which consist of periodic task models where the tasks can be released asynchronously. Job response time equations developed in this paper, for a given task arrival pattern, are used in bounding the preemptions. We then modify schedulability conditions for RM and EDF schedulers to account for preemption overheads, using these preemption bounds. Simulation results show that the bounds developed in this paper are tighter than previously known preemption bounds by a factor of upto 90%. *We also provide an estimate for the number of preemptions and show using simulations that this estimate is very close to the actual number of preemptions.*

1 Introduction

Correctness of a real-time system depends not only on logical correctness but also on timeliness. A real-time system is said to be schedulable according to a scheduling algorithm, if the timing constraints of workloads within the system can be always satisfied under the scheduling algorithm with given available resources. Schedulability analysis, which determines whether or not a system is schedulable, has been one of the most fundamental research problems in the real-time systems community. Since Liu and Layland [13] showed the optimality of the earliest deadline first (EDF) and rate-monotonic (RM) algorithms for dynamic- and fixed-priority preemptive uni-processor scheduling, schedulability analysis has been extensively studied for these two algorithms [13, 10, 3, 2, 5]. A common drawback shared by those previous studies [13, 10, 3, 2, 5] is that their techniques do not incorporate preemption overheads into schedulability analysis.

In preemptive processor scheduling, the scheduler may suspend the execution of a less urgent task and give the processor to a more urgent task. During preemption, the execution state or context of the preempted task must be stored for later use and the context of the preempting task must be loaded. We use the term *preemption overhead* to mean the time required to complete this overhead of work to be done during preemption. This preemption overhead must be accounted for in the schedulability analysis of real-time systems because it consumes real-time resource.

In addition to schedulability analysis, preemptions are also important in computing cache miss delays. For real-time systems that use instruction and data caches, cache miss delay adversely affects the worst case execution time estimates for real-time tasks. Bounding this delay is then important in providing timing assurances for the system. Since caches are shared among real-time tasks, task preemptions increase the cache miss delay. When a different task begins execution due to preemption, some of the data in the caches will become irrelevant. Hence,

to accurately compute the cache miss delay within a time interval, it is then necessary to know the number of preemptions occurring in that interval.

The periodic task model [13] is a well-known deterministic real-time task model. With its various extension, this model characterizes accurately many hard real-time applications, such as avionics and digital control where accurate control requires continual sampling and processing of data. In this paper, we assume a periodic real-time task system consisting of periodic, independent tasks with deadline equal to period. The task system is assumed to be scheduled using either RM or EDF scheduler. Further, each real-time task is assumed to be released with an initial release phase, and periodically thereafter based on the "as soon as" arrival pattern [13]. Asynchronous release of tasks in a task system can be characterized using the initial release phase of tasks. As will be seen later in the paper, bounding the number of preemptions in a task system requires computation of response times of jobs. For a periodic task model, these response times can be computed in a simple manner using iterative equations. Task dependences complicate response time computation because a task can be released only after all the tasks on which this task is dependent finish execution. Since, in this paper we focus on deriving equations for bounding the number of preemptions, we have assumed independent tasks. Dependent tasks can be considered in our model by using task offsets, which can be bounded using task dependences, in response time equations. Restricting the deadline to be same as period simplifies presentation of the paper; equations given in this paper can be easily modified to model tasks with arbitrary deadlines.

In this paper, we present upper as well as lower bounds on the number of preemptions occurring in a time interval for a given real-time task system. We have developed response time equations that compute response time of jobs given a task arrival pattern. We provide approximations for the response time of a job in order to avoid simulating the task system. Depending upon assumptions on the interference of higher priority workload released prior to the release time of a job, we have developed upper and lower bounds for the response time of the job. These response time bounds are then used to compute the bounds on the number of preemptions. Simulation results show that for a variety of real-time task systems, the preemption bounds derived in this paper are tighter than any known bounds. We have considered four task system parameters for the simulation experiments; utilization, number of tasks, scheduler (RM or EDF) and utilization distribution skew. We also provide an estimate for the number of preemptions by using only the lower bound for the response times of jobs. Simulation results show that this estimate is very close to the actual number of preemptions. We have then modified schedulability tests for RM and EDF schedulers using the preemption bounds developed in this paper, to account for preemption overheads incurred by the task system.

Related Work. Bounding the number of preemptions plays an important role in the analysis of real-time systems, particularly, in the analysis involving preemption, such as schedulability analysis [13], worst-case response time computations [11, 17, 15], cache miss delay computations [16, 14] etc. Loose upper bounds have often been used in these analyses in the past [8, 9, 6, 12, 1, 16, 14]. Under the assumption that every release of a job will result in a preemption, a very loose upper bound for the number of preemptions has been used for worst-case response time analysis under RM scheduling [8, 6] and cache miss delay computation [9, 16]. Estimated preemption bounds have also been used for bounding the cache miss delay in data caches [14]. Their estimate is tighter than the aforementioned upper bound on an average, but is not an upper bound. In [9] a bound on the number of preemptions is obtained by formulating an optimization problem for maximizing the number of preemptions under constraints of total number of released tasks. This will again give a loose upper bound as in [16]. In this paper, we present upper bounds on the number of preemptions for task systems that use RM or EDF schedulers, and our upper bound is tighter than previously known bounds.

Any real-time system analysis involving preemption can be made more precise with our preemption bounding technique. For example, in a compositional framework using periodic resource model abstractions, preemption overheads at each level can increase the resource demand of the system [1, 12, 7]. Hence it is necessary to bound preemption overheads in the schedulability analysis of such hierarchical systems. In [12], the authors have formulated the schedulability analysis problem for hierarchical systems, taking into account preemption overheads. In this formulation, they assume that the preemption cost over time is known for hierarchical systems without specifying how to obtain this cost. Upper bound on the number of preemptions developed in this paper can be used in their formulation to bound the preemption overhead incurred by the system.

The rest of the paper is organized as follows. Section 2 gives the system model and Section 3 derives the response time bounds for jobs. Section 4 develops equations to bound the number of preemptions in a task system using the response time bounds. Section 5 modifies RM and EDF schedulability conditions to account for preemption

overheads. Section 6 describes results comparing the preemption bounds developed in this paper with the existing bounds as well as with the actual number of preemptions generated using simulation. Section 7 concludes the paper and discusses future work. Bounding the number of preemptions plays an important role in the analysis of real-time systems, particularly, in the analysis involving preemption, such as schedulability analysis [13], worst-case response time computations [11, 17, 15], cache miss delay computations [16, 14] etc. Loose upper bounds have often been used in these analyses in the past [8, 9, 6, 12, 1, 16, 14]. Under the assumption that every release of a job will result in a preemption, a very loose upper bound for the number of preemptions has been used for worst-case response time analysis under RM scheduling [8, 6] and cache miss delay computation [9, 16]. Estimated preemption bounds have also been used for bounding the cache miss delay in data caches [14]. Their estimate is tighter than the aforementioned upper bound on an average, but is not an upper bound. In [9] a bound on the number of preemptions is obtained by formulating an optimization problem for maximizing the number of preemptions under constraints of total number of released tasks. This will again give a loose upper bound as in [16]. In this paper, we present upper bounds on the number of preemptions for task systems that use RM or EDF schedulers, and our upper bound is tighter than previously known bounds.

Any real-time system analysis involving preemption can be made more precise with our preemption bounding technique. For example, in a compositional framework using periodic resource model abstractions, preemption overheads at each level can increase the resource demand of the system [1, 12, 7]. Hence it is necessary to bound preemption overheads in the schedulability analysis of such hierarchical systems. In [12], the authors have formulated the schedulability analysis problem for hierarchical systems, taking into account preemption overheads. In this formulation, they assume that the preemption cost over time is known for hierarchical systems without specifying how to obtain this cost. Upper bound on the number of preemptions developed in this paper can be used in their formulation to bound the preemption overhead incurred by the system.

2 System Model

In this paper, we consider a periodic real-time task system that consists of a periodic task set scheduled by earliest deadline first (EDF) or rate-monotonic (RM) scheduler [13]. The periodic task system $\mathcal{TS}$ is characterized by a tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, such that $\mathcal{T}$ denotes a set of periodic tasks T_k, i.e., $\mathcal{T} = \{T_1, \cdots, T_n\}$, and $\mathcal{A}$ denotes the scheduling algorithm for the task set $\mathcal{T}$, where $\mathcal{A} = EDF/RM$. Each periodic task T_k is defined by $T_k = (p_k, e_k, \phi_k)$, where p_k is the period, e_k is the worst-case execution time, and ϕ_k is the phase ($0 \leq \phi_k < p_k$), such that the first release of T_k occurs at ϕ_k and each subsequent release of T_k occurs periodically with a separation of p_k between consecutive releases. We assume that each periodic task T_k is independent and preemptable at any time instant. Further, the relative deadline for each task T_k is assumed to be p_k.

Each release of task T_k is called a job. Let $J_{k,t}$ denote a job of task T_k whose release occurs at time instant t. The start time of a job $J_{k,t}$ is the length of the interval from its release instant to the time it begins execution. The response time of a job $J_{k,t}$ is the time taken by the job to finish e_k units of its execution starting from its release time. We will use the following notations for jobs.

- $R(J_{k,t})$: the release time of job $J_{k,t}$, which is t,
- $D(J_{k,t})$: the deadline of job $J_{k,t}$, which is $t + p_k$,
- $ST(J_{k,t})$: the start time of job $J_{k,t}$,
- $RT(J_{k,t})$: the response time of job $J_{k,t}$,
- $PJ_k(t')$: the most recently or previously released job $J_{k,t}$ of task T_k prior to time instant t', where $t' \geq \phi_k + 1$. That is, $PJ_k(t')$ represents the job $J_{k,t}$ such that no job of task T_k is released in the interval (t, t'). We will use the notation $PJ_k(t')$ to denote a job $J_{k,t}$ of T_k which can execute in the interval $[t' - 1, t')$. We can compute such t as follows:

$$t = \lfloor \frac{(t' - \phi_k - 1)}{p_k} \rfloor p_k + \phi_k.$$

In preemptive scheduling, the execution of a less urgent job can be suspended in order to execute a more urgent job. When the more urgent job completes, the less urgent job can resume its execution.

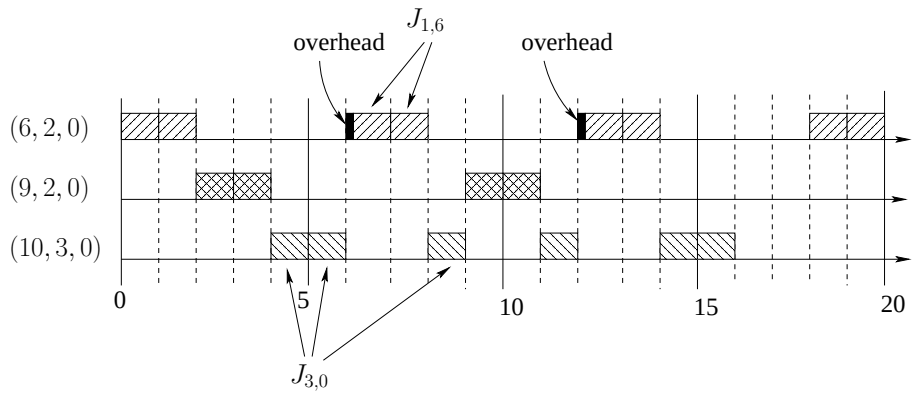

Figure 1. Preemption Overhead

Observation 2.1 *In preemptive priority-based scheduling, a preemption occurs at time instant t' if and only if*

1. *a job $J_{i,t'}$ of task T_i is released at time instant t' ($J_{i,t'}$ exists),*
2. *a job $J_{k,t}$ of task T_k was executing in the time interval $[t'-1, t')$ and did not finish its execution at time t', and*
3. *$J_{i,t'}$ has a higher priority than $J_{k,t}$.*

Every preemption will incur an execution overhead which is required for saving the context of the preempted job as well as for loading the context of the preempting job. As shown in Figure 1, assuming the tasks are scheduled using RM scheduler, job $J_{1,6}$ will incur a preemption overhead for storing the context of the preempted lower priority job $J_{3,0}$, as well as for loading the initial context of job $J_{1,6}$ itself.

As an upper bound for the number of preemptions in a task system, existing real-time analyses [8, 9, 6, 16] use a very loose bound which is the number of released jobs. For the task system $\mathcal{TS} = \langle \{T_1 = (6, 2, 0), T_2 = (9, 2, 0), T_3 = (10, 3, 0)\}, RM \rangle$ shown in Figure 1, these analyses would use an upper bound of 9 for the number of preemptions in the time interval $[0, 20]$ (9 is the number of jobs released in this interval). As can be seen in the figure, this is indeed a very loose upper bound because the actual number of preemptions occurring in this interval is only 2.

3 Job Response Time

The release of a job $J_{i,t'}$ will result in a preemption if and only if all the conditions given in Observation 2.1 in Section 2 are satisfied. Condition 2 in Observation 2.1 requires that a job having priority lower than $J_{i,t'}$ must execute in the interval $[t'-1, t')$. To determine whether a particular job is executing in some time interval for a given task system, we can either simulate the execution of the task system or analytically compute the time intervals in which the job executes. In this paper we will use job response time equations to determine whether a particular job executes in some interval. A job $J_{k,t}$ released before t' and having response time greater than t' will execute in the interval $[t'-1, t')$, if the response times of all jobs released before t' and having priority higher than $J_{k,t}$ less than or equal to $t'-1$. In this section we will derive response time and start time equations for jobs scheduled using RM and EDF schedulers.

3.1 Response Time Approximations

Worst case response time of a task given a periodic task system can be computed using equations given in [11, 17] for RM scheduler and in [15] for EDF scheduler. In this paper we are interested in deriving similar equations for the response time and start time of a job. Since the response time of a job can be much smaller than the worst case response time of the corresponding task, using job response times for computing preemption bounds will in general

result in tighter bounds. Computing the exact response time or start time of a job $J_{k,t}$ requires accurate knowledge of the interference of higher priority jobs released before t on the execution of $J_{k,t}$. In order to avoid task system simulation, we will approximate this interference caused by higher priority workload released before time t. The best-load response time is obtained by assuming zero interference from higher priority jobs released before time t and the worst-load response time is obtained by assuming maximum possible interference from higher priority jobs released before time t. The ***best-load response time*** of a job $J_{k,t}$ is the response time of the job obtained by assuming that jobs $PJ_l(t)$ that have priority higher than $J_{k,t}$ complete execution before t. The ***worst-load response time*** of a job $J_{k,t}$ is the response time of the job obtained by assuming that jobs $PJ_l(t)$ that have priority higher than $J_{k,t}$ execute as much as possible after time t without missing deadlines. Best-load start time and worst-load start time of jobs can be described similarly.

Figure 2 shows a schedule for a task system $\mathcal{TS} = \langle\{T_1 = (3,1,0), T_2 = (5,1,0), T_3 = (7,1,0), T_4 = (10,2,0)\}, RM\rangle$. In this schedule, interference from higher priority jobs $J_{1,9}, J_{2,5}$ and $J_{3,7}$ on the execution of job $J_{4,10}$ is assumed to be 0, i.e., jobs $J_{1,9}, J_{2,5}$ and $J_{3,7}$ have all finished execution by time 10. Similarly, Figure 3 shows a schedule for a task system $\mathcal{TS} = \langle\{T_1 = (3,1,0), T_2 = (5,1,0), T_3 = (7,1,0), T_4 = (10,2,0)\}, RM\rangle$ where maximum interference is assumed for the execution of job $J_{4,10}$ from higher priority jobs $J_{1,9}, J_{2,5}$ and $J_{3,7}$. In the figure, jobs $J_{1,9}$ and $J_{3,7}$ are assumed to execute completely after time 10 while job $J_{2,5}$ is assumed to finish execution before time 10. It is not possible for job $J_{2,5}$ with relative deadline 10 to execute after time 10 without missing its deadline.

3.2 Job Response Time under RM Scheduler

In this section we will derive response time equations for a task system that uses RM scheduler. Let $\mathcal{TS} = \langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, RM\rangle$ be the task system and let the priorities of these tasks be such that task T_i has higher priority than task T_j if and only if $i < j$. Let $BRT_{RM}(J_{k,t})$ denote the best-load response time, as described in Section 3.1, of a job $J_{k,t}$ in task system $\mathcal{TS}$. Iterative equation for computing $BRT_{RM}(J_{k,t})$ is then given by,

$$
\begin{aligned}
BRT^{(0)}_{RM}(J_{k,t}) &= e_k \\
BRT^{(i)}_{RM}(J_{k,t}) &= \sum_{l<k}\{(\lfloor\frac{(BRT^{(i-1)}_{RM}(J_{k,t}) + t - 1 - \phi_l)}{p_l}\rfloor - \lfloor\frac{(t-1-\phi_l)}{p_l}\rfloor)e_l\} + e_k
\end{aligned} \tag{1}
$$

Response time $BRT_{RM}(J_{k,t})$ is given by that value of $BRT^{(i)}_{RM}(J_{k,t})$ which satisfies the condition $BRT^{(i)}_{RM}(J_{k,t}) = BRT^{(i-1)}_{RM}(J_{k,t})$. $BRT^{(0)}_{RM}(J_{k,t})$ in Eq. (1) assumes zero interference from higher priority load released before time t. The iterative equation then computes the higher priority load in the current execution window of job $J_{k,t}$, such that the load is released some time at or after time t. Since schedulability condition requires that the response time of job $J_{k,t}$ be less than or equal to $D(J_{k,t}) = t + p_k$, computation of $BRT_{RM}(J_{k,t})$ takes time $O(np_k)$, assuming $\mathcal{TS}$ is schedulable. As shown in Figure 2, assuming zero interference from higher priority jobs $J_{1,9}, J_{2,5}$ and $J_{3,7}$ on the execution of job $J_{4,10}$, we get $BRT_{RM}(J_{4,10}) = 4$. Iterative equation for the best-load start time, $BST_{RM}(J_{k,t})$, can be obtained by substituting $e_k = 0$ in Eq.(1).

We now similarly give iterative equations for the worst-load response time of a job scheduled using RM scheduler. Let $WRT_{RM}(J_{k,t})$ denote the worst-load response time, as described in Section 3.1, of a job $J_{k,t}$ in task set $\mathcal{TS}$. For each job $PJ_l(t)$ having priority higher than $J_{k,t}$, let $e_{max}(PJ_l(t))$ denote the maximum interference that this job can cause to the execution of $J_{k,t}$. Then,

$$
e_{max}(PJ_l(t)) = \max\{0, \min\{e_l, \lceil\frac{(t-\phi_l)}{p_l}\rceil p_l + \phi_l - t - \sum_{m<l}(e_{max}(PJ_m(t)) + (k+1)e_m)\}\} \tag{2}
$$

where k satisfies the condition, $\lceil(t-\phi_m)/p_m\rceil p_m + \phi_m + kp_m \leq \lceil(t-\phi_l)/p_l\rceil p_l + \phi_l$. $e_{max}(PJ_l(t))$ is given by the minimum of e_l and the size of execution window $[t, D(PJ_l(t))]$ taking into consideration the interference caused to the execution of $PJ_l(t)$ by higher priority load executing in the window. The term $e_{max}(PJ_m(t)) + (k+1)e_m$ in Eq. (2) gives this interference caused by higher priority jobs of task T_m in the window $[t, D(PJ_l(t))]$. Iterative equation for $WRT_{RM}(J_{k,t})$ is then given by,

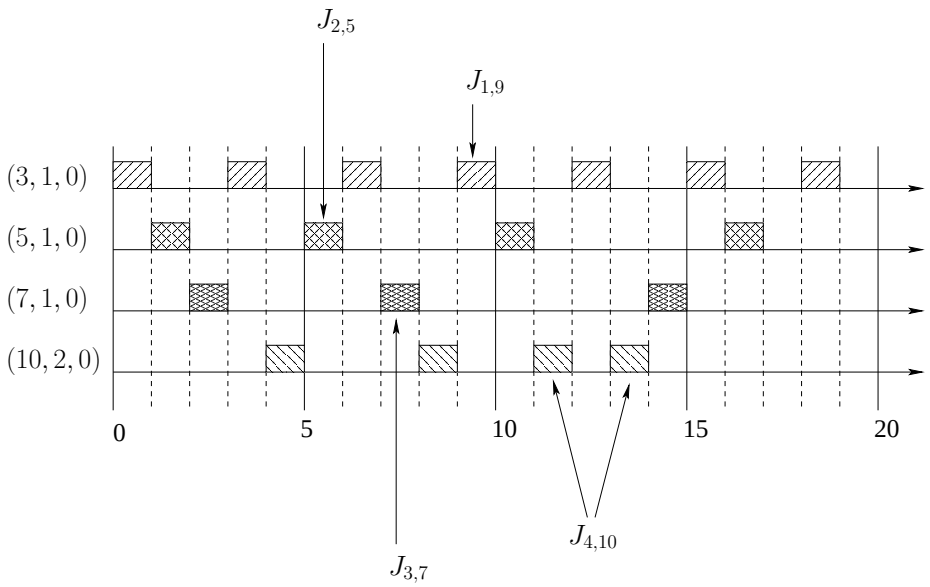

Figure 2. RM: Best-load Response Time

$$WRT^{(0)}_{RM}(J_{k,t}) = e_k + \sum_{l<k} e_{max}(PJ_l(t))$$

$$WRT^{(i)}_{RM}(J_{k,t}) = \sum_{l<k}\{(\lfloor\frac{(WRT^{(i-1)}_{RM}(J_{k,t}) + t - 1 - \phi_l)}{p_l}\rfloor - \lfloor\frac{(t-1-\phi_l)}{p_l}\rfloor)e_l\} + WRT^{(0)}_{RM}(J_{k,t}) \quad (3)$$

Response time $WRT_{RM}(J_{k,t})$ is given by that value of $WRT^{(i)}_{RM}(J_{k,t})$ which satisfies the condition $WRT^{(i)}_{RM}(J_{k,t}) = WRT^{(i-1)}_{RM}(J_{k,t})$. $WRT^{(0)}_{RM}(J_{k,t})$ in Eq. (3) assumes maximum interference from higher priority jobs released before time t, i.e., for each higher priority job $PJ_l(t)$, $WRT^{(0)}_{RM}(J_{k,t})$ adds its maximum interference $e_{max}(PJ_l(t))$ to the response time of $J_{k,t}$. $WRT_{RM}(J_{k,t})$ can be computed in time $O(n^2 + np_k)$ where $O(n^2)$ time is required to compute the maximum interference for all the higher priority jobs. As shown in Figure 3, assuming maximum interference to the execution of job $J_{4,10}$ from higher priority jobs $J_{1,9}, J_{2,5}$ and $J_{3,7}$, we get $WRT_{RM}(J_{4,10}) = 20$. Iterative equation for the worst-load start time, $WST_{RM}(J_{k,t})$, can be similarly derived by substituting $e_k = 0$ in Eq.(3).

3.3 Job Response Time under EDF Scheduler

In this section we will derive job response time equations for a task system that uses EDF scheduler. Let $\mathcal{TS} = \langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, EDF\rangle$ denote a task system. As per the EDF scheduling policy, job $J_{i,t}$ has higher priority than job $J_{j,t'}$ if and only if $D(J_{i,t}) < D(J_{j,t'})$. To break ties between jobs having the same deadline, we assume that a job $J_{i,t}$ has higher priority than another job $J_{j,t'}$ having the same deadline as job $J_{i,t}$ if and only if $i < j$. Let $BRT_{EDF}(J_{k,t})$ denote the best-load response time of a job $J_{k,t}$ in the task system $\mathcal{TS}$. The iterative equation for $BRT_{EDF}(J_{k,t})$ is then given as,

$$BRT^{(0)}_{EDF}(J_{k,t}) = e_k$$

$$BRT^{(i)}_{EDF}(J_{k,t}) = \sum_{l\neq k} \max\{0, (-\lfloor\frac{(t-1-\phi_l)}{p_l}\rfloor + \lfloor\frac{(BRT^{(i-1)}_{EDF}(J_{k,t}) + t - 1 - \phi_l)}{p_l}\rfloor - I_l)\}e_l + e_k \quad (4)$$

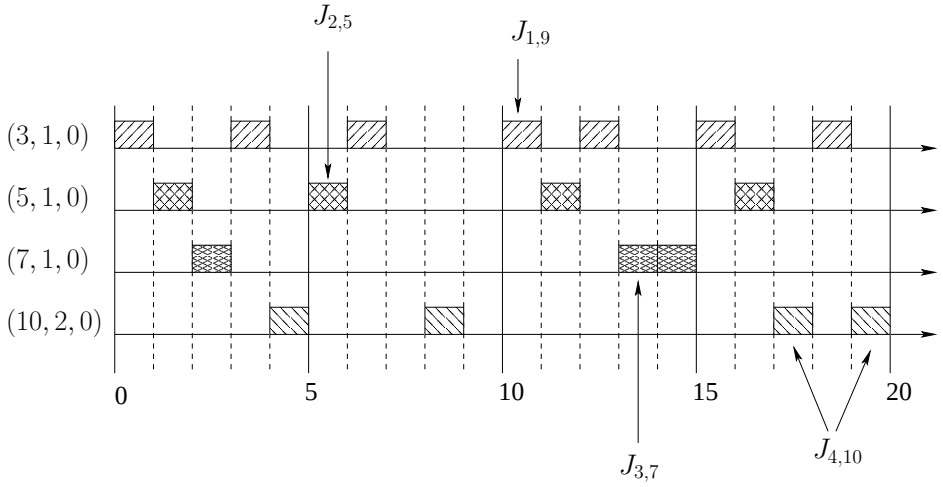

Figure 3. RM: Worst-load Response Time

where I_l is an indicator variable which takes value 1 if and only if the priority of the last job of task T_l considered in the computation of $BRT^{(i)}_{EDF}(J_{k,t})$ in Eq. (4) is lower than the priority of job $J_{k,t}$.

$$I_l = \begin{cases} 0 & \text{If } (\lceil \frac{(BRT^{(i-1)}_{EDF}(J_{k,t})+t-\phi_l)}{p_l} \rceil) p_l + \phi_l < D(J_{k,t}) \text{ or} \\ & ((\lceil \frac{(BRT^{(i-1)}_{EDF}(J_{k,t})+t-\phi_l)}{p_l} \rceil) p_l + \phi_l = D(J_{k,t})) \wedge (l < k) \\ 1 & \text{Otherwise} \end{cases} \quad (5)$$

Response time $BRT_{EDF}(J_{k,t})$ is given by that value of $BRT^{(i)}_{EDF}(J_{k,t})$ for which $BRT^{(i)}_{EDF}(J_{k,t}) = BRT^{(i-1)}_{EDF}(J_{k,t})$. Given the current execution window $[t, BRT^{(i-1)}_{EDF}(J_{k,t})]$ of job $J_{k,t}$, the iterative equation given by Eq. (4) computes the load of all higher priority jobs released within this window. $BRT_{EDF}(J_{k,t})$ can be computed in time $O(np_k)$. The best-load start time, $BST_{EDF}(J_{k,t})$, can be similarly obtained by substituting $e_k = 0$ in Eq.(4).

Worst-load response time $WRT_{EDF}(J_{k,t})$ and worst-load start time $WST_{EDF}(J_{k,t})$, can be derived using Eqs. (2), (3) and (5). We assume that the job indices for the job set $\{PJ_i(t)|i \neq k\}$ are rearranged based on their priorities in order to make the presentation simpler, i.e., we assume indices are rearranged such that the priority of job $PJ_m(t)$ is less than the priority of job $PJ_n(t)$ if and only if $m < n$. Then the maximum interference, $e_{max}(PJ_l(t))$, of the higher priority job $PJ_l(t)$, to the execution of job $J_{k,t}$ is given by Eq. (2) in Section 3.2. Iterative equation for $WRT_{EDF}(J_{k,t})$ is then given as,

$$WRT^{(0)}_{EDF}(J_{k,t}) = e_k + \sum_{l<k} \{e_{max}(PJ_l(t))\}$$

$$WRT^{(i)}_{EDF}(J_{k,t}) = \sum_{l \neq k} \max\{0, (-\lfloor \frac{(t-1-\phi_l)}{p_l} \rfloor + \lfloor \frac{(WRT^{(i-1)}_{EDF}(J_{k,t}) + t - 1 - \phi_l)}{p_l} \rfloor - I_l)\} e_l + WRT^{(0)}_{EDF}(J_{k,t}) \quad (6)$$

Indicator variable I_l is given by Eq. (5) in Section 3.2 where $BRT^{(i-1)}_{EDF}(J_{k,t})$ is replaced with $WRT^{(i-1)}_{EDF}(J_{k,t})$. Response time $WRT_{EDF}(J_{k,t})$ is given by that value of $WRT^{(i)}_{EDF}(J_{k,t})$ which satisfies the condition $WRT^{(i)}_{EDF}(J_{k,t}) = WRT^{(i-1)}_{EDF}(J_{k,t})$. $WRT_{EDF}(J_{k,t})$ can be computed in time $O(n^2 + np_k)$.

4 Preemption Bounds for Real-Time Task Systems

In this section, we assume that the best-load and worst-load, response and start times for a job $PJ_k(t')$ are all zero for values of t' less than $(\phi_k + 1)$, i.e., for values of t' where $PJ_k(t')$ is not defined. Also, we will refer to the response time of a job added to its release time as the absolute response time and the start time of a job added

to its release time as the absolute start time of the job. Conditions for the occurrence of a preemption given in Observation 2.1 indicate that for a preemption to occur at time instant t',

1. A job $J_{i,t'}$ of task T_i must be released at time t'
2. The maximum over absolute response times of all jobs $PJ_k(t'+1)$, that have priority higher than $J_{i,t'}$, must be less than t', i.e., no higher priority job was executing in the interval $[t'-1, t')$ or must not be released at time t'
3. The maximum over absolute response times of all jobs $PJ_k(t')$, that have priority lower than $J_{i,t'}$, must be greater than t', i.e., some lower priority job must finish execution after t',
4. The absolute start time of job $PJ_k(t')$ for which the maximum in condition 3 was achieved must be less than t'.

Best-load and worst-load, response and start times of jobs for a task system that uses RM or EDF scheduler can be computed using equations described in Section 3. In this section we will use these job response times to derive upper as well as lower bounds for the number of preemptions in a given time interval.

4.1 Preemption Upper Bound under RM Scheduler

An upper bound on the number of preemptions in a time interval can be obtained by counting the number of jobs released in this interval that will never cause a preemption. A job $J_{i,t'}$ will never cause a preemption if and only if any one of the following conditions is satisfied.

1. The maximum over best-load absolute response times of all jobs $PJ_k(t'+1)$ having priority higher than $J_{i,t'}$ is greater than t'. If some higher priority job released on or before time t' does not complete execution until after t' in the best-load case, then under all circumstances this job will complete execution after time t'. Then, job $J_{i,t'}$ cannot cause a preemption in this case.
2. No job $PJ_k(t')$, having priority lower than $J_{i,t'}$, has best-load absolute start time less than t' and worst-load absolute response time greater than t'. If all the lower priority jobs either begin execution after t' in the best-load case or finish execution before t' in the worst-load case, then none of them can execute in the interval $[t'-1, t')$. In this case as well, job $J_{i,t'}$ will not cause a preemption.

Any job which does not satisfy all of the above conditions can cause a preemption and hence will contribute towards the upper bound. Let $\langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, RM\rangle$ denote a task system $\mathcal{TS}$ where task T_i has higher priority than task T_j if and only if $i < j$. We now derive an equation for the upper bound on the number of preemptions for task system $\mathcal{TS}$. For a job $J_{i,t'}$ in the task system $\mathcal{TS}$, the maximum over best-load absolute response times of all jobs $PJ_k(t'+1)$ having priority higher than $J_{i,t'}$ is given as,

$$HRT_{J_{i,t'}} = \max_{k<i}\{BRT_{RM}(PJ_k(t'+1)) + R(PJ_k(t'+1))\} \tag{7}$$

Let $CT_{PJ_k(t')}$ be equal to the worst-load absolute response time of job $PJ_k(t')$ provided its best-load absolute start time is less than t', and zero otherwise. $CT_{PJ_k(t')}$ is then given as,

$$CT_{PJ_k(t')} = \begin{cases} 0 & BST_{RM}(PJ_k(t')) + R(PJ_k(t')) \geq t' \\ WRT_{RM}(PJ_k(t')) + R(PJ_k(t')) & \text{Otherwise} \end{cases} \tag{8}$$

The maximum over worst-load absolute response times of all jobs $PJ_k(t')$, having best-load start time less than t' and priority lower than $J_{i,t'}$, is given as,

$$LRT_{J_{i,t'}} = \max_{k>i}\{CT_{PJ_k(t')}\}$$

where $CT_{PJ_k(t')}$ is given by Eq. (8). An upper bound on the number of preemptions can now be given for the task system $\mathcal{TS}$ using $HRT_{J_{i,t'}}$ and $LRT_{J_{i,t'}}$ given in Eqs. (7) and (9) respectively.

Definition 4.1 (Preemption Upper Bound) *Let $PU_{\mathcal{TS}}(L)$ denote an upper bound on the number of preemptions occurring within time interval $(0, L)$ for the task system $\mathcal{TS}$. Let $JR(L) = \{0 \leq t_1, t_2, \cdots, t_m \leq L\}$ denote the set of time instants at which jobs of task system $\mathcal{TS}$ are released in the interval $(0, L)$. Further, for each time instant t', let $JS(t')$ denote the set of jobs released at time t'. Then,*

$$PU_{\mathcal{TS}}(L) = \sum_{t' \in JR(L)} \sum_{J_{i,t'} \in JS(t')} I_{J_{i,t'}} \tag{9}$$

where $I_{J_{i,t'}}$ is an indicator variable which will have value 1 if and only if $J_{i,t'}$ can cause a preemption.

$$I_{J_{i,t'}} = \begin{cases} 0 & \text{If } HRT_{J_{i,t'}} \geq t \text{ or } LRT_{J_{i,t'}} \leq t \\ 1 & \text{Otherwise} \end{cases} \tag{10}$$

Response time computations take $O(np_k)$ for each higher priority task and $O(n^2 + 2np_k)$ for each lower priority task. The computation time for $PU_{\mathcal{TS}}(L)$ is then $O(L((n^2 + 2n \max_k\{p_k\}) \times n)) = O(L(n^3 + n^2 \max_k\{p_k\}))$.

Theorem 4.2 (Preemption Upper Bound) *$PU_{\mathcal{TS}}(L)$, given by Definition 4.1, is an upper bound on the number of preemptions occurring within time interval $(0, L)$ for the task system $\mathcal{TS}$.*

Proof $HRT_{J_{i,t'}}$ gives the maximum best-load absolute response time over all jobs $PJ_k(t'+1)$ having priority higher than $J_{i,t'}$. Since the actual response time of any higher priority job must be greater than $HRT_{J_{i,t'}}$, $HRT_{J_{i,t'}} \geq t'$ implies that at least one higher priority job will finish execution after t'. Hence $J_{i,t'}$ will start execution after all the higher priority jobs have finished execution, which means no lower priority job will be executing when $J_{i,t'}$ starts execution. Hence job $J_{i,t'}$ will never cause a preemption in this case.

$LRT_{J_{i,t'}}$ gives the maximum worst-load absolute response time over all jobs $PJ_k(t')$ having priority lower than $J_{i,t'}$. In the computation of $LRT_{J_{i,t'}}$, only those lower priority jobs are considered such that their best-load absolute start time is less than t'. Since the actual response time of these lower priority jobs will be less than $LRT_{J_{i,t'}}$, $LRT_{J_{i,t'}} \leq t'$ implies no lower priority job which starts execution before t' will execute until after t'. In this case as well, job $J_{i,t'}$ will never cause a preemption.

This means that $I_{J_{i,t'}}$ in Eq. (10) is 0 if and only if job $J_{i,t'}$ in the task system $\mathcal{TS}$ will not cause a preemption. Thus we have shown that $PU_{\mathcal{TS}}(L)$ computes an upper bound on the number of preemptions occurring in the interval $(0, L)$ for $\mathcal{TS}$.

□

For example, consider the job $J_{2,5}$ for the task system $\langle\{T_1 = (3, 1, 0), T_2 = (5, 1, 0), T_3 = (7, 1, 0), T_4 = (10, 2, 0)\}, RM\rangle$ shown in Figure 4. The higher priority job $J_{1,3}$ has a best-load absolute response time of 4. The lower priority job $J_{3,0}$ has a worst-load absolute response time of 3 and $J_{4,0}$ has a worst-load absolute response time of 9. Both the lower priority jobs start execution before time 5 in the best-load case. Hence the release of job $J_{2,5}$ may preempt job $J_{4,0}$ thereby contributing to the preemption upper bound. Similarly, it can be shown that the release of job $J_{1,3}$ in Figure 4 will never cause a preemption and hence it will not contribute to the upper bound.

4.2 Preemption Lower Bound under RM Scheduler

A lower bound on the number of preemptions occurring within a time interval can be obtained by counting the number of jobs released in this interval that will surely cause a preemption. A job $J_{i,t'}$ will surely cause a preemption if and only if in the worst case, no higher priority job will execute in the interval $[t' - 1, t')$ and some lower priority job with absolute response time greater than t' will surely execute in the interval $[t' - 1, t')$. In other words, a job $J_{i,t'}$ will surely cause a preemption if and only if,

1. the maximum over worst-load absolute response times of all jobs $PJ_k(t' + 1)$ having priority higher than $J_{i,t'}$ is less than t'

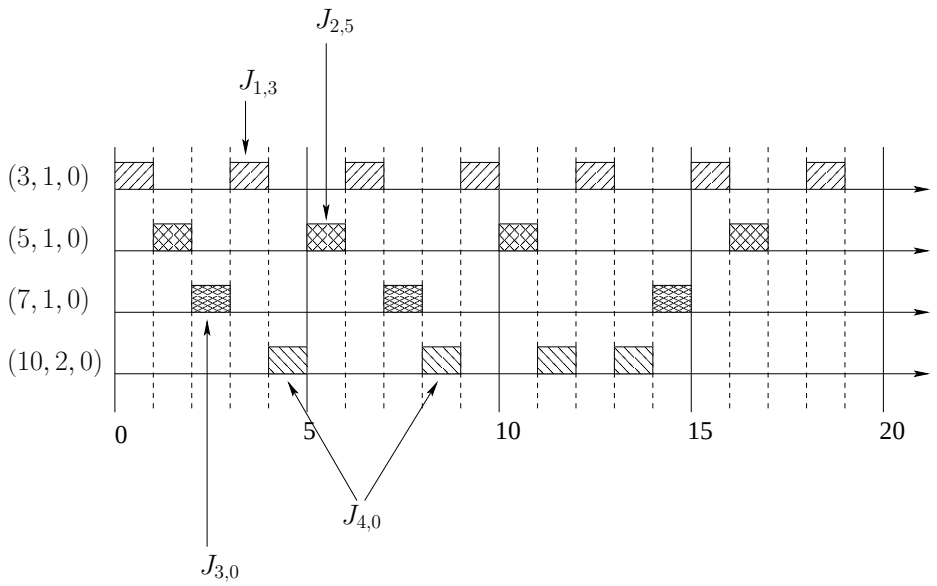

Figure 4. RM: Preemption Upper Bound

2. the best-load absolute response time of some job $PJ_j(t')$, having worst-load absolute start time less than or equal to t' and priority lower than $J_{i,t'}$, is greater than or equal to t'
3. the worst-load absolute response time of all jobs $PJ_k(t')$ having priority higher than $PJ_j(t')$ but lower than $J_{i,t'}$ is less than t'

We now derive the equations for computing the lower bound on the number of preemptions for a task system $\mathcal{TS} = \langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, RM\rangle$. We will give these equations without discussing them because they are similar to the equations derived for the upper bound in Section 4.1. For a job $J_{i,t'}$, the maximum over worst-load absolute response times of all jobs $PJ_k(t')$ having priority higher than $J_{i,t'}$ is given by,

$$HRT_{J_{i,t'}} = \max_{k<i}\{WRT_{RM}(PJ_k(t'+1)) + R(PJ_k(t'+1))\}$$

Similarly, the maximum over best-load absolute response times of all jobs $PJ_k(t')$ having priority lower than $J_{i,t'}$ is given by,

$$LRT_{J_{i,t'}} = \max_{k>i}\{CT_{PJ_k(t')}\}$$

where $CT_{PJ_k(t')}$ is 0 if any one of the following conditions is satisfied.

- $PJ_k(t')$ does not start execution until time instant t' in the worst-load case,
- $PJ_{k-1}(t')$ has worst-load absolute response time greater than or equal to t'. In this case, there is a chance that $PJ_k(t')$ might not execute in the interval $[t'-1, t')$.
- $CT_{PJ_{k-1}(t')} = 0$. This means that some higher priority task may execute in the interval $[t'-1, t')$ and hence $PJ_k(t')$ cannot surely execute in that interval.

In all other cases $CT_{PJ_k(t')}$ is equal to the best-load completion response time of job $PJ_k(t')$. Assuming $CT_{PJ_i(t')} = -1$ we get,

$$CT_{PJ_k(t')} = \begin{cases} 0 & WST_{RM}(PJ_k(t')) + R(PJ_k(t')) \geq t' \text{ or} \\ & CT_{PJ_{k-1}(t')} = 0 \text{ or} \\ & WRT_{RM}(PJ_{k-1}(t')) + R(PJ_{k-1}(t')) \geq t' \\ BRT_{RM}(PJ_k(t')) + R(PJ_k(t')) & \text{Otherwise} \end{cases}$$

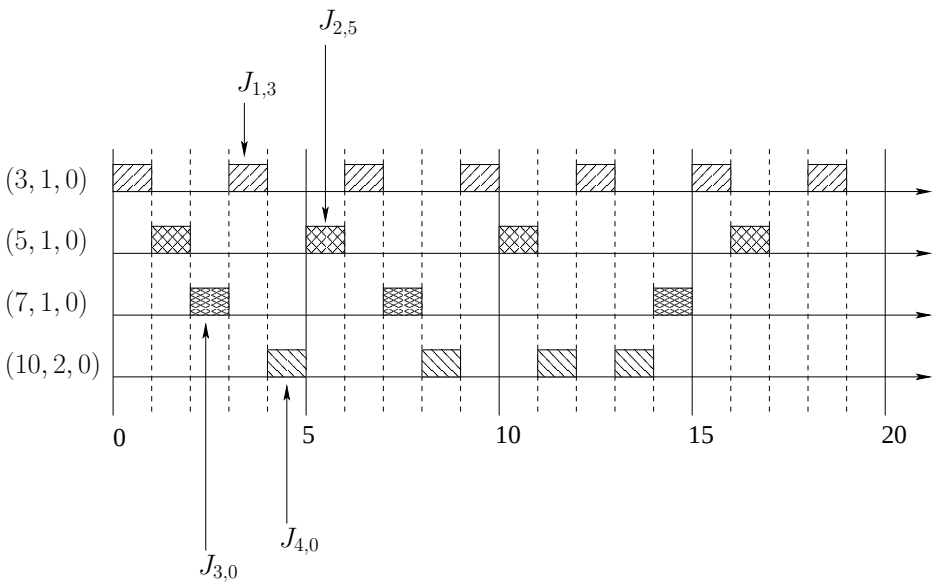

Figure 5. RM: Preemption Lower Bound

Definition 4.3 (Preemption Lower Bound) *Let $PL_{\mathcal{TS}}(L)$ denote the lower bound on the number of preemptions occurring within the time interval $(0, L)$ for task system $\mathcal{TS}$. Let $JR(L) = \{0 \leq t_1, t_2, \cdots, t_n \leq L\}$ denote time instants in the interval $(0, L)$ when jobs in $\mathcal{TS}$ are released. Also, let $JS(t')$ denote the set of jobs in $\mathcal{TS}$ released at time instant t'. Then,*

$$PL_{\mathcal{TS}}(L) = \sum_{t' \in JR(L)} \sum_{J_{i,t'} \in JS(t')} I_{J_{i,t'}} \tag{11}$$

$I_{J_{i,t'}}$ is an indicator variable with value 1 if and only if job $J_{i,t'}$ will surely result in a preemption. $I_{J_{i,t'}}$ is given by Eq. (10) in Section 4.1.

Response time computations take $O(n^2 + np_k)$ for each higher priority task and $O(2n^2 + 3np_k)$ for each lower priority task. Since these computations are repeated at each job release, the total computation time for $PL_{\mathcal{TS}}(L)$ is $O(L((n^2 + np_k) \times n)) = O(L(n^3 + n^2 p_k))$.

For example, consider the job $J_{2,5}$ for the task system $\langle\{T_1 = (3, 1, 0), T_2 = (5, 1, 0), T_3 = (7, 1, 0), T_4 = (10, 2, 0)\}, RM\rangle$ shown in Figure 5. The higher priority job $J_{1,3}$ has a worst-load absolute response time of 4. The lower priority job $J_{3,0}$ has best-load absolute response time of 3 and $J_{4,0}$ has best-load absolute response time of 9. Further, job $J_{3,0}$ finishes execution before time instant 5 even in the worst-load case. Hence release of job $J_{2,5}$ will surely preempt job $J_{4,0}$ and thus will contribute to the lower bound.

Theorem 4.4 (Preemption Lower Bound) *$PL_{\mathcal{TS}}(L)$ given in Definition 4.3 is a lower bound on the number of preemptions occurring within time interval $(0, L)$ for the task system $\mathcal{TS}$.*

Proof Similar to the proof of Theorem 4.2 given in Section 4.1.

□

4.3 Preemption Bounds under EDF Scheduler

Bounds on the number of preemptions occurring within a time interval for a task system that uses EDF scheduler can be computed using equations similar to the ones given in Sections 4.1 and 4.2. Let $\langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, EDF\rangle$ be a task system $\mathcal{TS}$. Let $PU_{\mathcal{TS}}(L)$ denote the upper bound and let $PL_{\mathcal{TS}}(L)$ denote the lower bound on the number of preemptions occurring within time interval $(0, L)$ for $\mathcal{TS}$. Also, for each job $J_{i,t'}$ released in the interval $(0, L)$, let $\mathcal{HP}_{J_{i,t'}}$ denote a subset of the set of jobs $\{PJ_k(t' + 1) | k \neq i\}$ having priority higher than $J_{i,t'}$. Similarly, let $\mathcal{LP}_{J_{i,t'}}$ denote an ordered subset of the set of jobs $\{PJ_k(t') | k \neq i\}$ having priority

lower than $J_{i,t'}$. We assume that all the lower priority jobs $\{PJ_k(t')|PJ_k(t') \in \mathcal{LP}_{J_{i,t'}}\}$ are assigned fresh indices based on their priorities, i.e., job $PJ_m(t') \in \mathcal{LP}_{J_{i,t'}}$ has higher priority than job $PJ_n(t') \in \mathcal{LP}_{J_{i,t'}}$ if and only if $m < n$. $PU_{\mathcal{TS}}(L)$ is then given by Definition 4.1 in Section 4.1, where all the response time computations are done using response time equations for EDF schedulers given in Section 3.3. Similarly, $PL_{\mathcal{TS}}(L)$ is given by Definition 4.3 in Section 4.2, where all the response time computations are done using response time equations for EDF schedulers given in Section 3.3. Also, $HRT_{J_{i,t'}}$ is computed using the set of jobs $\mathcal{HP}_{J_{i,t'}}$ and $LRT_{J_{i,t'}}$ is computed using the ordered set of jobs $\mathcal{LP}_{J_{i,t'}}$.

4.4 Estimate for Number of Preemptions

In this section we will modify the preemption upper bound computation given in Section 4.1 to generate an estimate for the number of preemptions. We will generate this estimate by using only best-load job response and start times in the upper bound equations given in Section 4.1. Let $PE_{\mathcal{TS}}(L)$ denote this estimate for the number of preemptions occurring in the interval $(0, L)$ for a real-time task system $\mathcal{TS} = \langle \{T_1, \cdots, T_n\}, \mathcal{A} \rangle$. Consider the worst-load absolute response time, $CT_{PJ_k(t')}$, given in Eq. (8). Using the best-load response time for jobs we get,

$$CT_{PJ_k(t')} = \begin{cases} 0 & BST_{RM}(PJ_k(t')) \\ & +R(PJ_k(t')) \geq t' \\ BRT_{RM}(PJ_k(t')) & \\ +R(PJ_k(t')) & \text{Otherwise} \end{cases} \tag{12}$$

The preemption estimate, $PE_{\mathcal{TS}}(L)$, is then given by Definition. (4.1), where $LRT_{J_{i,t}}$ given in Eq. (9) uses $CT_{PJ_k(t')}$ in Eq. (12). Simulation results given in Section 6 indicate that this preemption estimate is very close to the actual number of preemptions, for a variety task systems.

5 Schedulability Analysis for RM and EDF Schedulers

In this section we derive conditions for schedulability of a real-time task system under RM or EDF scheduler. These schedulability conditions will take into consideration the execution overhead incurred by the system as a result of preemptions. Prior to the start of execution of a real-time job the context for the job must be initialized. We assume that the execution overhead incurred by the task system for initialization of context of a job is included in the worst case execution time of the job itself. Hence we will only consider preemption overheads when deriving the modified schedulability tests in this section.

5.1 Job Response Time with Preemption Overhead

Best-load as well as worst-load, response and start times given in Section 3 do not take into account the execution overhead incurred by the task system as a result of preemptions. Let δ_1 be the execution overhead incurred by a real-time job that preempts another lower priority job in the system. Since a preemption will increase the execution time of the preempting job, the interference of this job in the response time of other lower priority jobs in the system will also increase. Thus for schedulability analysis with preemption overhead, this increased interference must be incorporated in the response time equations. Also, to determine whether the release of a job $J_{i,t'}$ leads to a preemption, we compute the response and start times of all jobs $PJ_k(t'+1)$ having priority higher than $J_{i,t'}$ and all jobs $PJ_k(t')$ having priority lower than $J_{i,t'}$ as given in Section 4. We are only interested in the response time and start time of these jobs relative to time instant t', i.e., we are only interested in knowing whether these response and start times are greater than, equal to or less than t'. Since, while determining whether a preemption occurred at time instant t', we have complete information about preemptions occurring at all time instants upto t', we can use this knowledge to bound the preemption overhead in the computation of response and start times. This means that in order to compute the response time or start time of a job $J_{k,t}$ for determining whether the release of job $J_{i,t'}$ leads to a preemption, we can upper bound the preemption overhead in the interval (t, t') using preemption information computed earlier. In Figure 4, to determine whether the release of job $J_{1,3}$ will result in a preemption, we need to compute the response and start times of the lower priority job $J_{4,0}$. While computing this

response and start times, there is complete knowledge of the number of preemptions in the interval $[0, 3)$ which can be used to bound the preemption overhead in this interval.

Let $PU^k_{\mathcal{TS}}(t, t_2)$ denote an upper bound on the number of preemptions in the time interval $[t, t_2)$ such that these preemptions are caused by jobs having priority higher than job $J_{k,t}$. While computing the response time or start time of $J_{k,t}$, we only need to consider jobs that have priority higher than $J_{k,t}$ for preemption overhead. Let the response time of job $J_{k,t}$ be computed to determine whether the release of job $J_{i,t'}$ leads to a preemption. We now give the modified best-load response time equation for $J_{k,t}$ in the task system $\mathcal{TS} = \langle\{T_1, \cdots, T_n\}, RM\rangle$. We assume that a task T_i has higher priority than a task T_j in $\mathcal{TS}$ if and only if $i < j$.

$$
\begin{aligned}
BRT^{(0)}_{RM}(J_{k,t}) =& e_k + PU^k_{\mathcal{TS}}(t, \min\{e_k, t'\})\delta_1 \\
BRT^{(i)}_{RM}(J_{k,t}) =& \sum_{l<k}\{(\lfloor\frac{(BRT^{(i-1)}_{RM}(J_{k,t}) - 1 + t - \phi_l)}{p_l}\rfloor - \lfloor\frac{(t - 1 - \phi_l)}{p_l}\rfloor)e_l\} + e_k \\
BRT^{(i)}_{RM}(J_{k,t}) =& BRT^{(i)}_{RM}(J_{k,t}) + PU^k_{\mathcal{TS}}(t, \min\{BRT^{(i)}_{RM}(J_{k,t}), t'\})\delta_1
\end{aligned} \tag{13}
$$

In Eq. (13), preemption overhead $PU^k_{\mathcal{TS}}(t, \min\{BRT^{(i)}_{RM}(J_{k,t}), t'\})\delta_1$ is added to the current response time $BRT^{(i)}_{RM}(J_{k,t})$ of the job. The best-load response time $BRT_{RM}(J_{k,t})$ with preemption overhead is given by that value of $BRT^{(i)}_{RM}(J_{k,t})$ such that $BRT^{(i)}_{RM}(J_{k,t}) = BRT^{(i-1)}_{RM}(J_{k,t})$. Other job response time equations in Section 3 can be modified similarly to account for preemption overheads.

5.2 Schedulability Analysis for RM Scheduler

In this section we will derive schedulability conditions for a task system that uses RM scheduler. These conditions will account for the execution overhead incurred by the task system due to preemptions. Schedulability test for a task system scheduled using RM scheduler is done using worst case task response time analysis [13]. Let $\mathcal{TS} = \langle\{T_1 = (p_1, e_1, 0), \cdots, T_n = (p_n, e_n, 0)\}, RM\rangle$ denote a task system where all tasks are released synchronously at time $t = 0$. We assume that a task T_i has higher priority than a task T_j if and only if $i < j$. The worst case response time $r(k)$ of a task T_k can be computed using iterative equation given in [13].

$$
\begin{aligned}
r^{(0)}(k) &= e_k \\
r^{(i)}(k) &= \sum_{l<k}\lceil r^{(i-1)}(k)/p_l\rceil e_l + e_k
\end{aligned} \tag{14}
$$

$r(k)$ is given by that value of $r^{(i)}(k)$ for which $r^{(i)}(k) = r^{(i-1)}(k)$. It has been shown in [13] that synchronous release of tasks is indeed the critical instant for all the tasks in the task system. Schedulability condition for $\mathcal{TS}$ is then given as,

$$\forall k : 1 \leq k \leq n, \exists t \in (0, p_k], r(k) = t$$

To account for preemption overheads in the schedulability analysis of the task system, we will now modify Eq. (14) using preemption bounds derived in Section 4. We assume that these preemption bounds have been derived using response time equations that account for preemption overheads, as described in Section 5.1. Let $\mathcal{TS} = \langle\{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, RM\rangle$ denote a task system where we assume that a task T_i has higher priority than a task T_j if and only if $i < j$. For each task T_k, let the initial release phase shift values $\phi_1, \cdots, \phi_k$ be given by the critical instant for task T_k in the task system $\mathcal{TS}$ and let δ_1 be the execution overhead for each job preemption.

Definition 5.1 *Let $r_p(k)$ denote the worst case response time of a task T_k such that $r_p(k)$ accounts for the execution overhead incurred by the task system as a result of preemptions. Then,*

$$
\begin{aligned}
r^{(0)}_p(k) &= e_k \\
r^{(i)}_p(k) &= \sum_{l<k}\lceil\frac{(r^{(i-1)}_p(k) - \phi_l)}{p_l}\rceil e_l + e_k \\
r^{(i)}_p(k) &= r^{(i)}_p(k) + PU_{\mathcal{TS}}(r^{(i)}_p(k))\delta_1
\end{aligned} \tag{15}
$$

The worst case response time $r_p(k)$ for task T_k is given by that value of $r_p^{(i)}(k)$ for which $r_p^{(i)}(k) = r_p^{(i-1)}(k)$. In the computation of $PU_{\mathcal{TS}}(r_p^{(i)}(k))$ we only consider tasks having priority higher than T_k, i.e., $PU_{\mathcal{TS}}(r_p^{(i)}(k))$ computes an upper bound on the number of preemptions in a time interval of size $r_p^{(i)}(k)$ when the task system only consists of tasks $T_1, \cdots, T_k$. In each iteration, Eq. (15) first computes the worst case response time of task T_k without considering preemption overheads. This worst case response time is then modified with the execution overhead incurred by the task system for preemptions. Modified schedulability conditions for $\mathcal{TS}$ is then given as,

$$\forall k : 1 \leq k \leq n, \exists t \in (0, p_k], r_p(k) = t \tag{16}$$

5.3 Schedulability Analysis for EDF Scheduler

In this section we will derive schedulability conditions for a task system that uses EDF scheduler taking into consideration preemption overheads. Schedulability test for a task system scheduled with EDF scheduler is given using the demand bound function for that system. The demand bound function $dbf(t)$ for a task system, is the worst case resource demand generated by this system in any time interval of length t. Let $\mathcal{TS} = \langle \{T_1 = (p_1, e_1, 0), \cdots, T_n = (p_n, e_n, 0)\}, EDF \rangle$ denote a synchronous task system. The demand bound function for $\mathcal{TS}$ as given in [4] is,

$$dbf(t) = \sum_{k=1}^{n} (\lfloor t/p_k \rfloor e_k) \tag{17}$$

Assuming LCM denotes the least common multiple of the periods of all the tasks in $\mathcal{TS}$, schedulability condition for $\mathcal{TS}$ as given in [4] is,

$$\forall t \in (0, LCM], dbf(t) \leq t \tag{18}$$

The demand bound function $dbf(t)$ given in Eq. (17) can be modified to account for the execution overhead incurred by the task system as a result of preemptions. Let $\mathcal{TS} = \langle \{T_1 = (p_1, e_1, \phi_1), \cdots, T_n = (p_n, e_n, \phi_n)\}, EDF \rangle$ denote a task system. We assume that the initial release phase shift values $\phi_1, \cdots, \phi_n$ are given by the critical instant for the task system. Let δ_1 denote the execution overhead incurred by the task system for each job preemption. Demand bound function with preemption overhead, $dbf_p(t)$, is then given as,

$$dbf_p(t) = \sum_{k=1}^{n} (\lfloor \max\{0, (t - \phi_k)\}/p_k \rfloor e_k) + PU_{\mathcal{TS}}(t)\delta_1$$

For each time interval of length t, we add the preemption overhead $PU_{\mathcal{TS}}(t)\delta_1$ incurred by the task system to its demand bound function. We assume that the preemption bound, $PU_{\mathcal{TS}}(t)$, is derived using response time equations that account for preemption overheads, as described in Section 5.1. Modified schedulability condition taking into account preemption overhead is then given by,

$$\forall t \in (0, LCM], dbf_p(t) \leq t$$

6 Evaluation

In this section we evaluate the proposed preemption upper bound in comparison with previously known preemption bounds as well as with the exact number of preemptions generated using simulation. For a given task system $\mathcal{TS} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a time interval $[0, L)$, we compute the previously known preemption bound, $PB_{\mathcal{TS}}(L)$, which is the number of jobs released in the interval $[0, L)$, our preemption bound $PU_{\mathcal{TS}}(L)$ computed using Eq. (9) and our preemption estimate $PE_{\mathcal{TS}}(L)$ computed as in Section 4.4. We also simulate the task system to obtain the actual number of preemptions, $PN_{\mathcal{TS}}(L)$, occurring in the given interval.

For simulations, we use the following parameters for the task system $\mathcal{TS}$: n is the number of tasks in $\mathcal{TS}$, U is the utilization of $\mathcal{TS}$, and s is the utilization distribution skew; the utilization of each task is determined such that the task with the lowest priority has an utilization of $s \cdot U$ and the remaining utilization of $(1 - s) \cdot U$ is then uniformly distributed among all the other tasks. For each set of these parameter values, we execute 100 task sets

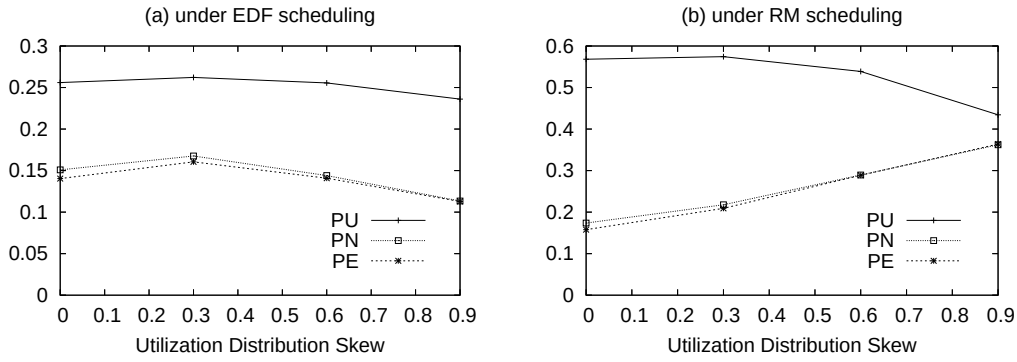

Figure 6. Effect of utilization distribution skew

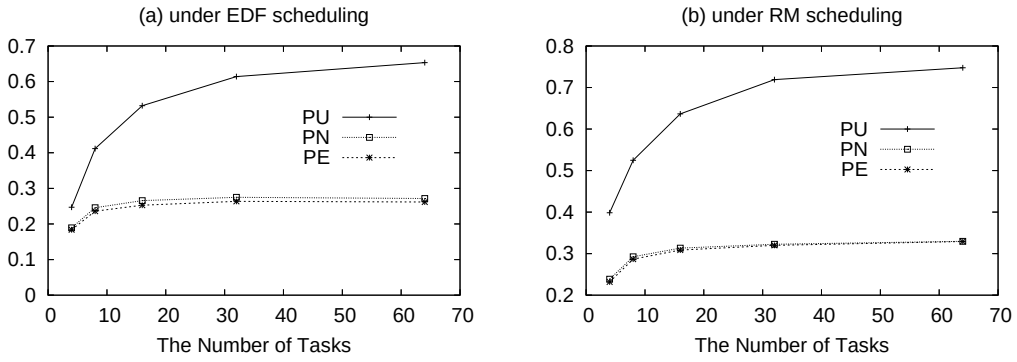

Figure 7. Effect of the number of tasks

$\mathcal{T}$ having different values for the period and execution time of tasks. We simulate the same task set $\mathcal{T}$ twice, once under RM and the other time under EDF scheduling.

In Figures 6, 7 and 8, the plots labeled "PU", "PE" and "PN", show the normalized values $PU_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$, $PE_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$ and $PN_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$, respectively. Each point in the "PU" and "PE" plots represents the mean value over the 100 different task sets, for the normalized preemption bound $PU_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$ and the estimate $PE_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$, respectively. Similarly, each point in the "PN" plots represents the mean value for the normalized actual number of preemptions $PN_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$ generated by simulation.

Figure 6 shows the effect of utilization distribution skew of a task system on the preemption upper bound $PU_{\mathcal{TS}}(L)$. For this simulation, we have assumed that the number of tasks $n = 10$ and utilization $U = 0.5$. In the figure, when s is 0, it means that every task has the same task utilization. The preemption bound does not vary much with respect to the actual number of preemptions for increasing skew values under EDF scheduling. Under RM scheduling, the preemption bound infact gets better as the skew increases. As can be observed in the figure, the preemption estimates $PE_{\mathcal{TS}}(L)$ are very close to the actual number of preemptions irrespective of the skew. This observation for the estimate holds for all the parameter values, as can be seen in Figures 7 and 8.

Figure 7 shows the effect of the number of tasks on the preemption bound under RM and EDF scheduling, where $U = 0.5$ and $s = 0.5$. The figure shows that both our preemption bound and the actual number of preemptions increase rapidly for a small task set, and then stabilize for larger values of n.

Figure 8 shows the effect of utilization on our preemption upper bound under RM and EDF scheduling. These simulations have been done with $n = 10$ and $s = 0.5$. The figure shows that our preemption bound improves substantially for increasing values of utilization under EDF scheduling, but under RM scheduling, the bound increases with utilization.

The implications of our simulation results can be summarized as follows: (1) our preemption upper bound

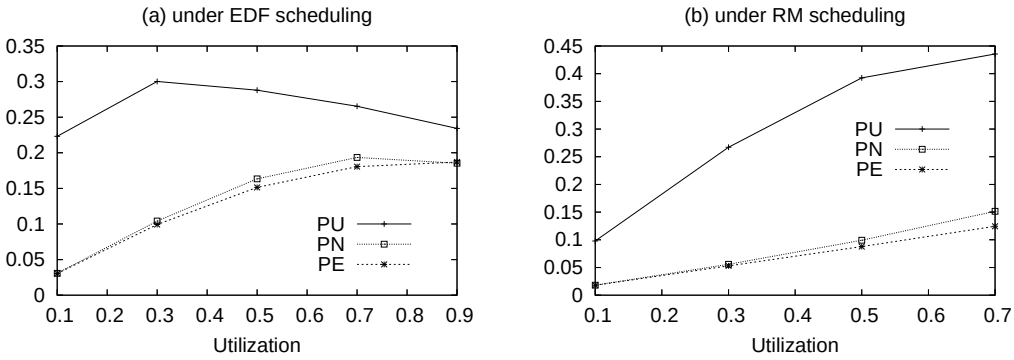

Figure 8. Effect of utilization

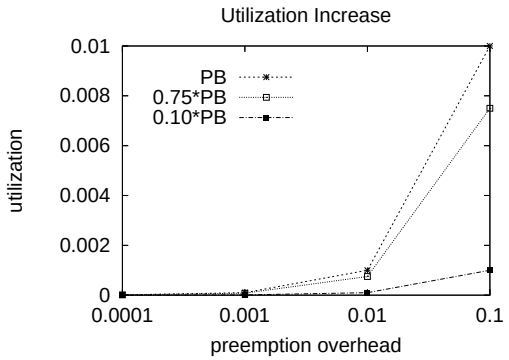

Figure 9. Utilization Increase due to Preemption Overhead

is much tighter than the previously known bound under both RM and EDF scheduling. (2) the bound does not deteriorate significantly for increasing values of utilization, skew or number of tasks. (3) goodness of the preemption estimate suggests that our approach of computing preemption bounds using job response times could lead to extremely tight bounds with further research.

Let α denote the normalized preemption upper bound $PU_{\mathcal{TS}}(L)/PB_{\mathcal{TS}}(L)$ and let δ be the execution overhead incurred by the task system for each job preemption. If the number of tasks in the system is n and the average value for the period of tasks is $p*$, then the increase in utilization due to preemption overhead using the known bound $PB_{\mathcal{TS}}(L)$ can be given as,

$$UI_{PB} = \frac{\frac{L}{p*} \times n \times \delta}{L} = \frac{n \times \delta}{p*}$$

Similarly, the increase in utilization due to preemption overhead using upper bound $PU_{\mathcal{TS}}(L)$ can be given as,

$$UI_{PU} = \frac{n \times \delta \times \alpha}{p*}$$

Figure 9 plots the increase in utilization of the task system for increasing values of preemption overhead δ, where $n = 10$ and $p* = 100$. That is, when $\delta = 0.1$, the utilization increase UI_{PB} is 0.01. As can be seen in the figure, for small values of δ, the difference in utilization increase is insignificant. But for larger values of δ, the upper bound developed in this paper can provide significant savings on utilization.For example, when $\alpha = 0.1$ and $\delta = 0.1$, the utilization increase $UI_{PU} = 0.001$ while the utilization increase $UI_{PB} = 0.01$. This gives an utilization saving of 0.009.

7 Conclusion

In this paper we have developed upper and lower bounds for the number of preemptions occurring within a time interval for a real-time task system that uses RM or EDF scheduler. These bounds have been developed using job response time bounds developed in this paper. Schedulability conditions under RM and EDF schedulers have also been modified to account for preemption overheads. The preemption bounds developed in this paper have been used to compute the worst case execution overhead incurred by the task set as a result of job preemptions. Simulation experiments show that the upper bound developed in this paper is tighter than previously known preemption upper bounds by a factor of upto 90%. This simulation has been done for a large set of task systems with varying parameters. Hence, any real-time analysis that uses preemption bounds can be made more precise by using the upper bound developed in this paper.

We are interested in determining the critical instants for task systems when preemption overheads are taken into consideration.

References

[1] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *Proc. of International conference on Embedded software*, pages 95–103, 2004.

[2] N. Audsley, A. Burns, and A. Wellings. Deadline monotonic scheduling theory and application. *Control Engineering Practice*, 1(1):71–78, 1993.

[3] S. Baruah, R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Journal of Real-Time Systems*, 2:301–324, 1990.

[4] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proc. of IEEE Real-Time Systems Symposium*, pages 182–190, December 1990.

[5] E. Bini and G. C. Buttazzo. The space of rate monotonic schedulability. In *Proc. of IEEE Real-Time Systems Symposium*, December 2002.

[6] A. Burns and A. Wellings. *Real-Time Systems and Programming Languages*. Addison Wesley Longmain, 2001.

[7] R. I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proc. of IEEE Real-Time Systems Symposium*, 2005.

[8] D. I. Katcher, H. Arakawa, and J. K. Strosnider. Engineering and analysis of fixed priority schedulers. *Software Engineering*, 19(9):920–934, 1993.

[9] C. Lee, J. Hahn, Y. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, and C. S. Kim. Analysis of cache-related preemption delay in fixed-priority preemtive scheduling. *IEEE Transactions on Computers*, 47(6):700–713, 1998.

[10] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Proc. of IEEE Real-Time Systems Symposium*, pages 166–171, 1989.

[11] J. P. Lehoczky, L. Sha, and J. K. Strosnider. Enhanced aperiodic responsiveness in hard real-time environments. In *Proc. of IEEE Real-Time Systems Symposium*, pages 110–123, 1992.

[12] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proc. of Euromicro Conference on Real-Time Systems*, 2003.

[13] C.L. Liu and J.W. Layland. Scheduling algorithms for multi-programming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[14] H. Ramaprasad and F. Mueller. Bounding preemption delay within data cache reference patterns for real-time tasks. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, 2006.

[15] M. Spuri. Analysis of deadline scheduled real-time systems. Technical Report RR-2772, INRIA, France, 1996.

[16] J. Staschulat, S. Schliecker, and R. Ernst. Scheduling analysis of real-time systems with precise modeling of cache related preemption delay. In *Proc. of Euromicro Conference on Real-Time Systems*, pages 41–48, 2005.

[17] K. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analysing fixed priority hard real-time tasks. *Real-Time Systems Journal*, 6(2), 1994.