Supporting Soft Real-Time Sporadic Task Systems on Uniform Heterogeneous Multiprocessors with No Utilization Loss

Guangmo Tong, *Student Member, IEEE,* and Cong Liu, *Member, IEEE*

Abstract—Uniform heterogeneous multicore architectures are becoming increasingly popular due to their potential of achieving high performance and energy efficiency compared to the homogeneous multicore architectures. In such systems, the real-time scheduling problem becomes more challenging because processors have different speeds. Prior research on uniform heterogeneous multiprocessor real-time scheduling has focused on hard real-time systems, where, significant processing capacity may have to be sacrificed in the worst-case to ensure that all deadlines are met. As meeting hard deadlines is overkill for many soft real-time systems in practice, this paper shows that on soft real-time uniform heterogeneous multiprocessors, bounded response times can be ensured for globally-scheduled sporadic task systems with no utilization loss. A GEDF-based scheduling algorithm, named as GEDF-H, is presented and response time bounds are established under both preemptive and non-preemptive GEDF-H scheduling. Extensive experiments show that the magnitude of the derived response time bound is reasonable, often smaller than four task relative deadlines. To the best of our knowledge, this paper is the first to show that soft real-time sporadic task systems can be supported on uniform heterogeneous multiprocessors without utilization loss under global scheduling, and with reasonable predicted response times.

Index Terms—scheduling, multiprocessor, real-time, modeling and prediction

1 Introduction

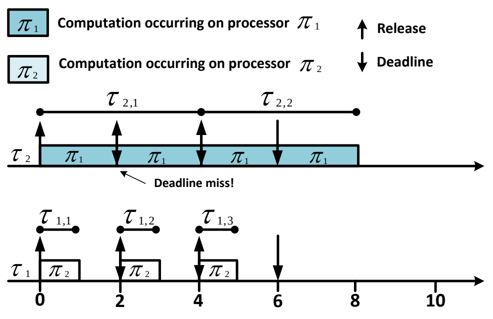
GIVEN the need to achieve higher performance without driving up power consumption and heat dissipation, most chip manufacturers have shifted to multicore architectures. An important subcategory of such architectures are those that are uniform heterogeneous in design. For example, the Intel's QuickIA platform [7] employs two kinds of processors with different speeds. By integrating processors with different speeds, such architectures can provide high performance and power efficiency [16]. Heterogeneous multicore architectures have been widely adopted in various domains, ranging from embedded applications (e.g. consumer multimedia [14] and video object tracking [13]) to high performance computing systems (e.g. GPGPU [4]). In order to apply heterogeneous multicore architectures in a real-time system, a fundamental problem is to analyze the schedulability of such systems.

Most prior work on supporting real-time workloads on such uniform heterogeneous multiprocessors has focused on hard real-time (HRT) systems. Unfortunately, if all task deadlines must be viewed as hard, significant processing capacity must be sacrificed in the worst-case, due to either inherent schedulability-related utilization loss—which is unavoidable under most scheduling schemes—or high runtime overheads—which typically arise in optimal schemes that avoid schedulability-related loss. In many systems where less stringent notions of real-time correctness suffice, such loss can be avoided by viewing deadlines as soft. In this paper, we consider the problem of scheduling soft real-time (SRT) sporadic task systems on a uniform heterogeneous multiprocessor; the notion of SRT correctness we consider is that response time is bounded [15].

All multiprocessor scheduling algorithms follow either a *partitioning* or *globally-scheduling approach* (or some combination of the two). Under partitioning, tasks are statically mapped to processors, while under global scheduling, they may migrate. Under partitioning schemes, constraints on overall utilization are required to ensure timeliness even for SRT systems due to bin-packing-related loss. On the other hand, a variety of global schedulers including the widely studied global earliest-deadline-first (GEDF) scheduling algorithm are capable of ensuring bounded response times for sporadic task systems on a homogeneous multiprocessor, as long as the system is not over-utilized [9]. Motivated by this optimal result, we investigate whether GEDF remains optimal in a uniform heterogeneous multiprocessor SRT system.

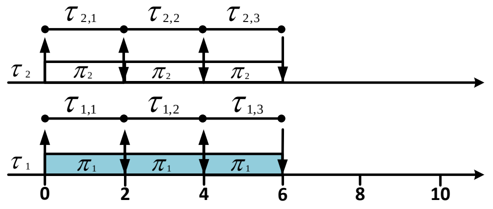
Under GEDF, we select m highest-priority jobs at any time instant and execute them on m processors. The job prioritization rule is according to earliest-

• *G. Tong and C. Liu are with the Department of Computer Science Erik Jonsson School of Engineering and Computer Science The University of Texas at Dallas 800 W. Campbell Road; MS EC31 Richardson, TX 75080 U.S.A.*
E-mail: {guangmo.tong; cong}@utdallas.edu



Computation occurring on processor π1
Computation occurring on processor π2
Release
Deadline
τ2,1
τ2,2
τ2
π1
Deadline miss!
τ1,1
τ1,2
τ1,3
τ1
π2
0
2
4
6
8
10

(a) Arbitrary processor selection



(b) Executing higher-utilization tasks on higher-speed processors

Fig. 1: Motivational example.

deadline-first. Regarding the processor selection rule (i.e., which processor should be selected for executing which job), it is typical to select processors in an arbitrary manner. On a homogeneous multiprocessor, such an arbitrary processor selection rule is reasonable since all processors have identical speeds. However, on a uniform heterogeneous multiprocessor, this arbitrary strategy may fail to schedule a SRT sporadic task system that is actually feasible under GEDF. Consider a task system with two sporadic tasks $\tau_1(2,2)$ and $\tau_2(4,2)$ (notation $\tau_i(C_i, T_i)$ denotes that task τ_i has an execution cost of C_i units, which costs C_i time units on a unit-speed processor, and a minimum inter-arrival time of T_i time units) scheduled on a uniform heterogeneous multiprocessor with two processors, π_1 with speed of one unit execution per unit time and π_2 with speed of two units execution per unit time. Assume in the example that task deadlines equal their minimum inter-arrival times and priority ties are broken in favor of τ_1. Fig. 1(a) shows the corresponding GEDF schedule with an arbitrary processor selection strategy for this task system. As seen in the figure, if we arbitrarily select processors for job executions, the response time of τ_2 grows unboundedly. However, if we define specific processor selection rules—for example always executing tasks with higher utilizations on processors with higher speeds—then this task system becomes schedulable, as illustrated in Fig. 1(b).

The above example suggests that *on a uniform heterogeneous multiprocessor, the processor selection strategy is critical to ensuring schedulability*. Motivated by this key observation, we consider in this paper whether it is possible to develop a GEDF-based scheduling algorithm with a specific processor selection rule, which can schedule SRT sporadic task systems on a uniform heterogeneous multiprocessor with no utilization loss.

Contribution. In this paper, we design and analyze a GEDF-based scheduling algorithm GEDF-H (GEDF for Heterogeneous multiprocessors) for supporting SRT sporadic task systems on a uniform heterogeneous multiprocessor that contains processors with different speeds. The analysis in this paper shows that under both preemptive and non-preemptive GEDF-H scheduling the bounded response times could be ensured if $U_{sum} \leq R_{sum}$ and Eq. (1) (in Sec. 3) holds, where U_{sum} is the total task utilization, R_{sum} is the total system capacity, and Eq. (1) is a requirement on the relationship between task parameters and processor parameters. Intuitively speaking, Eq. (1) implies that a system with a small number of high-speed processors cannot support too many high-utilization tasks even if the total resource capacity is sufficient. We show via a counterexample that task systems that violate Eq. (1) may have unbounded response time under any scheduling algorithm. As demonstrated by experiments, the derived response time bound under GEDF-H is reasonably low, often within four task relative deadlines.

Organization. This paper is organized as follows. In Sec. 2, we introduce the related work. Then in Sec. 3, we formally define the system model and the GEDF-H algorithm is described in Sec. 4. In Secs. 5, 6 and 7, we present the analysis of preemptive and non-preemptive GEDF-H scheduling. In Sec. 8, we show the experimental results. We conclude in Sec. 10.

2 Related Work

In the general heterogeneous multiprocessors, the rate of execution of a task depends on both the processor platform and the task system, and even not all tasks may be able to execute on all processors. In this paper, we limit our attention to uniform heterogeneous multiprocessors, where the rate of the execution of a task is directly proportional to the speed of the processor.

The real-time scheduling problem on uniform heterogeneous multiprocessors has received much attention.

For the HRT case, Funk *et al.* in [12] provides an exact test to determine whether a given periodic task system is feasible on a specified uniform multiprocessor. Partitioning approaches have been proposed in [1]–[3], [11], [16] and quantitative approximation ratios have been derived for quantifying the quality of these approaches. Unfortunately, such partitioning approaches inherently suffer from bin-packing-related utilization loss, which may be significant in many cases. The feasibility problem of global scheduling of HRT sporadic task systems on a uniform heterogeneous multiprocessor has also been studied [2]. In [8], a global scheduling algorithm has been implemented on Intel's QuickIA uniform heterogeneous prototype platform and experimental studies showed that this approach is effective in improving the system energy efficiency.

For the SRT case, most of the prior works have been focused on partitioning approach. A semi-partitioned approach has been proposed in [15], where tasks are categorized as either "fixed" or "intergroup" and processors are partitioned into groups according to their speeds. Tasks belonging to the fixed category are only allowed to migrate among processors within in the task's assigned group. Only tasks belonging to the migrating category are allowed to migrate among groups. Although this approach is quite effective in many cases, it yields utilization loss and requires several restricted assumptions (e.g., the system contains at least 4 processors and each processor group contains at least two processors). In [17], an improved partitioned approach is presented. Under this approach, the multiprocessor scheduling could be reduced to the uniprocessor case and the utilization loss can be eliminated. Unfortunately, few works have been focused on the global scheduling on uniform multiprocessors for SRT systems. Thus, our focus in this paper is on designing GEDF-based global schedulers that are able to ensure bounded response time with no utilization loss under both preemptive and non-preemptive scheduling.

Other prior works (e.g., [5], [6] and [10]) have also studied the problem of energy-efficient scheduling in multiprocessors with variant speeds. These works aim to minimize the energy consumption by either scaling voltage and frequency of processors, or manipulating the scheduling scheme of tasks. Different from those works, this paper focuses on improving the utilization of uniform heterogeneous multiprocessors under the SRT scheduling.

3 System Model

In this paper, we consider the problem of scheduling n sporadic SRT tasks on $m \geq 1$ uniform heterogeneous processors.

Let $\chi = \{\pi_1, ..., \pi_m\}$ denotes the set of m uniform heterogeneous processors and v_{π_i} be the speed of processor π_i. Without the loss of generality, we assume π_i is sorted by the speed of the processor in the non-increasing order. Assume there are $z \geq 1$ kinds of processors distinguished by their speeds. Let χ_i $(1 \leq i \leq z)$ and $M_i \geq 1$ denote the subset of the i^{th} kind of processors in χ and the number of processors in χ_i respectively. Thus, $\chi = \bigcup_{i=1}^{z} \chi_i$ and $m = \sum_{i=1}^{z} M_i$. We assume the processors in χ_1 have unit speed and processors in χ_i have speed α_i where $\alpha_i < \alpha_{i+1}, 1 \leq i \leq z-1$. For clarity, we use α_{max} to denote the maximum speed (i.e., $\alpha_{max} = \alpha_z = v_{\pi_1}$). Let $R_{sum} = \sum_{i=1}^{z} \alpha_i \cdot M_i$.

Let $\tau = \{\tau_1, ..., \tau_n\}$ denote the set of n independent sporadic tasks. We define the unit workload to be the amount of work done under the unit speed within a unit time. We assume that each job of τ_i executes for at most C_i workload which needs C_i time units under the unit speed. The j^{th} job of τ_i, denoted $\tau_{i,j}$, is released at time $r_{i,j}$ and has an absolute deadline at time $d_{i,j}$. Each task τ_i has a minimum inter-arrival time T_i, which specifies the minimum time between two consecutive job releases of τ_i, and a deadline D_i, which specifies the relative deadline of each such job, i.e., $d_{i,j} = r_{i,j} + D_i$. The utilization of a task τ_i is defined as $u_i = C_i/T_i$, and the utilization of the task system τ as $U_{sum} = \sum_{\tau_i \in \tau} u_i$. An sporadic task system τ is said to be an *implicit-deadline* system if $D_i = T_i$ holds for each τ_i. Due to space limitation, we limit attention to implicit-deadline sporadic task systems in this paper.

Successive jobs of the same task are required to execute in sequence. If a job $\tau_{i,j}$ completes at time t, then its *response time* is $t - r_{i,j}$. The response time of a task (resp. a task system) is the maximum response time of any of its jobs (resp. tasks). Note that, when a job of a task misses its deadline, the release time of the next job of that task is not altered. We require $u_i \leq \alpha_{max}$, and $U_{sum} \leq R_{sum}$, for otherwise the response time of a task system must grows unboundedly in the worst-case.

In this paper, we assume that the time system is continuous and the parameters are positive rational numbers.

Based on the system model, several terms are defined as follows.

Definition 1. Let $\overline{U}_k$ be the sum of the k largest u_i , $\overline{C}_k$ be the sum of the k largest C_i and $\overline{V}_k$ be the sum of the k smallest $u_i \cdot C_i$. Such values are used in the expression of the response time bound derived later in this paper.

Definition 2. A job is considered to be *completed* if it has finished its execution. We let $f_{i,j}$ denote the completion time of job $\tau_{i,j}$.

Definition 3. Job $\tau_{i,j}$ is *pending* at time t if $r_{i,j} < t < f_{i,j}$. Job $\tau_{i,j}$ is *enabled* at t if $r_{i,j} \leq t < f_{i,j}$, and its predecessor (if any) has completed by t. Note that each task has at most one enabled job at any time instant.

Definition 4. If an enabled job $\tau_{i,j}$ dose not execute at time t, then it is *preempted* at t.

Definition 5. A time instant t is *busy* (resp. *non-busy*) for a job set β if there exists (resp. does not exist) an $\varepsilon > 0$ that all m processors execute jobs in β during $(t, t + \varepsilon)$. A time interval $[a, b)$ is *busy* (resp. *non-busy*) for β if all (resp. not all) the instants within $[a, b)$ are busy for J.

The following property directly follows from the definition above.

Property 1. Suppose β is a set of jobs and t is a non-busy time instant for β. At most $m - 1$ tasks have jobs in β pending at t, for otherwise t would have to

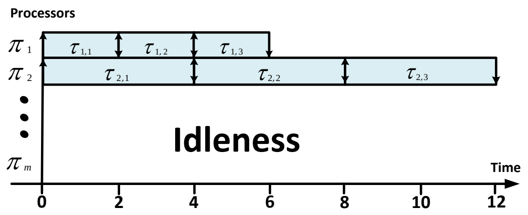


Fig. 2: GEDF schedule of the tasks in counterexample.

become busy.

On a uniform heterogeneous multiprocessor, the response time can still grow unboundedly, even if $u_i \leq \alpha_{max}$ and $U_{sum} \leq R_{sum}$ hold. This is illustrated by the following counterexample.

Counterexample. Consider a sporadic task system with two tasks $\tau_1 = \tau_2 = (2, 1)$ and a uniform heterogeneous multiprocessor with $m \geq 3$ processors where π_1 has a speed of $\alpha_{max} = 2$ and other $m-1$ processors have unit speed. For this system, $u_1 = u_2 = \alpha_{max} = 2$ and $R_{sum} = 2+(m-1) = m+1 \geq 4 = U_{sum}$. Note that, the ratio of U_{sum}/R_{sum} may approximate to 0 when m is arbitrarily large. However, as seen in the GEDF schedule illustrated in Fig. 2, regardless of the value we choose for m, the response time of τ_2 still grows unboundedly. Actually, we analytically prove that this task system cannot be scheduled under any global or partitioned schedule algorithm. This counterexample implies that a task system may not be feasible on a uniform heterogeneous multiprocessor even provided $U_{sum} \leq R_{sum}$. As seen in Fig. 2, adding more unit speed processors does not help because there are two tasks with utilization greater than 1 while only one processor with speed greater than 1. Motivated by this observation, we enforce the following requirement.

Let $\Phi_i = \{\tau_j | \alpha_i < u_j\}$, $1 \leq i < z$, and $|\Phi_i|$ be the number tasks in Φ_i. For completeness, let $\Phi_0 = \tau$. Let $\Psi_i = \bigcup_{j=i+1}^{z} \chi_j$, $0 \leq i < z$, and $|\Psi_i|$ be the number of processor in Ψ_i. Thus, Φ_i is the set of tasks that would fail their deadlines if run entirely on a processor of type i or lower, and Ψ_i is the set of processors of type i+1 or higher. For each $1 \leq i < z$, we require

$$|\Phi_i| \leq |\Psi_i| \tag{1}$$

Intuitively, Eq. (1) requires that *if we have k processors with speed larger than α_i, then at most k tasks with utilization larger than α_i can be supported in the system.* Note that, other than $U_{sum} \leq R_{sum}$, we do not place any restriction on U_{sum}. The following example illustrates Eq. (1).

Example 1. Consider a task system with 4 tasks, $\tau_1 = (2, 1), \tau_2 = (2, 1), \tau_3 = (1, 1), \tau_4 = (1, 1)$ and a uniform heterogeneous multiprocessor consisting of 3 processors, π_1, π_2 and π_3, where $v_{\pi_1} = v_{\pi_2} = 2.5$ and $v_{\pi_3} = 1$. Thus, $\chi_1 = \{\pi_3\}$, $\chi_2 = \{\pi_1, \pi_2\}$ and $\Psi_1 = \{\pi_1, \pi_2\}$. For this task system, $u_1 = u_2 = 2, u_3 = u_4 = 1$ and we have $\Phi_0 = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, $\Phi_1 = \{\tau_1, \tau_2\}$. Thus, we have $|\Phi_1| = 2 \leq |\Psi_1| = 2$. This system clearly meets the requirement stated in Eq. (1).

Algorithm 1 Preemptive GEDF-H

```
1: Input: An set $J^*$ of the enabled jobs at time instant $t$
2: Output: A mapping $f$ from $J^*$ to the processor set $\chi = \{\pi_1, ..., \pi_m\}$
3: $k \leftarrow \min(m, |J^*|)$;
4: Let $J'$ be the set of k highest priority jobs in $J^*$;
5: Sort the jobs in $J'$ by their utilizations in the non-increasing order. Let $J' = \{j_i, ..., j_k\}$;
6: for $i = 1 : k$ do
7:     $f(j_i) \leftarrow \pi_i$;
8: for each job $j'$ in $J^*$ but not in $J'$ do
9:     $f(j') \leftarrow null$;   // $j'$ is preempted by other jobs
10: Return $f$;
```

In the next section, we describe the GEDF-H scheduling algorithm.

4 GEDF-H Scheduling Algorithm

On a homogeneous multiprocessor, at any time instant, under GEDF, we can arbitrarily choose processors for tasks because processors have the same speed. However, on a uniform heterogeneous multiprocessor, if we arbitrarily choose processors for tasks, the bounded response time cannot be guaranteed as discussed in Sec. 1. Motivated by this key observation, we design a GEDF-based scheduling algorithm GEDF-H to support SRT sporadic task systems on a uniform heterogeneous multiprocessor.

Under GEDF-H, released jobs are prioritized by their absolute deadlines. We assume that ties are broken by task ID (lower IDs are favored). Thus, two jobs cannot have the same priority.

Preemptive GEDF-H. Under the preemptive case, at any time instant t, GEDF-H follows the following two phases to schedule a given task system on m uniform heterogeneous processors.

1) **Job selection phase.** Suppose there are n' enabled jobs at time instant t. GEDF-H selects $\mathrm{k} = \min(\mathrm{n}', \mathrm{m})$ enabled jobs, namely J_i, $1 \leq i \leq k$, with the highest priorities for execution. Assume these jobs J_i are sorted by their utilizations in the non-increasing order, where ties are broken by task ID (lower IDs are favored).
2) **Processor selection phase.** For $1 \leq i \leq n'$, GEDF-H assigns J_i to execute on π_i (π_i is sorted by the speed as mentioned in Sec. 3). In other words, GEDF-H assigns the higher-utilization job to the slowest processor.

Preemptive GEDF-H is formally shown in Algorithm 1.

Non-preemptive GEDF-H. Under the non-preemptive case, at any time instant t, GEDF-H follows the following two phases to schedule a given task system on m uniform heterogeneous processors.

1) **Job selection phase.** Let β_1 be the set of enabled jobs that are not executing on any processor at t,

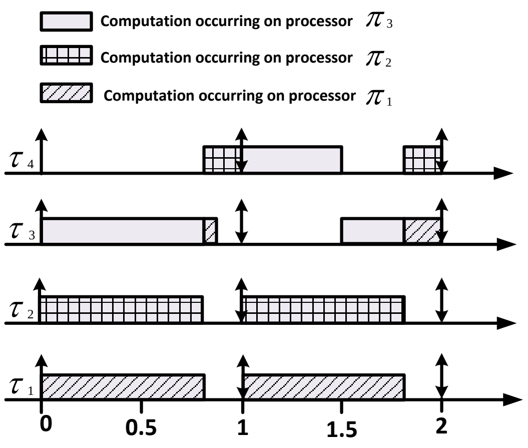

Computation occurring on processor π_3
Computation occurring on processor π_2
Computation occurring on processor π_1
τ_4
τ_3
τ_2
τ_1
0
0.5
1
1.5
2

Fig. 3: GEDF-H schedule of the tasks in Example 1.

and β_2 be the set of jobs that are executing at t. Assume the number of jobs in β_1 and β_2 are n_1 and n_2. Let $\mathrm{k} = \min\big(\mathrm{n_1},\ \max(0,\ \mathrm{m} - \mathrm{n_2})\big)$ and β_1^* be the set of k highest priority jobs in β_1. Then, GEDF-H selects $\beta_1^* \bigcup \beta_2$ for execution at t. Let J_i, $1 \le i \le k + n_2$, be the jobs in $\beta_1^* \bigcup \beta_2$ sorted by their utilization in the non-increasing order. By the selection of n_1, n_2, and k, $k + n_2 \le m$.

2) **Processor selection phase.** If $k + n_2 > 0$, i.e., $\beta_1^* \bigcup \beta_2$ is not empty, for $1 \le i \le k + n_2$, GEDF-H assigns J_i to execute on π_i.

Note that, GEDF-H is still a job-level static-priority scheduler because we do not change a job's priority at runtime. The processor selection phase of GEDF-H offers the following property.

Property 2. At any time instant t, if a job of task τ_i is executing on a processor $\pi^{'}$ with speed $v_{\pi'}$ where $\alpha_j < u_i \le \alpha_{j+1}$, we have $u_i \le v_{\pi'}$.

Proof: Let v_{τ_i} be the slowest speed of the processors on which jobs of τ_i could execute under GEDF-H. Thus, by processor selection rule of GEDF-H and Eq. (1), we have

$$v_{\tau_i} \ge \alpha_{j+1}$$

Therefore,

$$v_{\pi'} \ge v_{\tau_i} \ge \alpha_{j+1} \ge u_i \tag{2}$$

□

Fig. 3 shows the preemptive GEDF-H schedule of the task system in Example 1 within time interval $[0, 2)$. At time instant 1, in the job selection phase of GEDF-H, $\tau_{4,1}$, $\tau_{2,2}$ and $\tau_{1,2}$ are selected for execution; in the processor selection phase, $\tau_{4,1}$ is assigned to the slowest processor π_1.

In the Secs. 5, 6 and 7, we present our analysis of both preemptive GEDF-H and non-preemptive GEDF-H. Our analysis draws inspiration from [9], the seminal work for SRT scheduling on homogeneous multiprocessors. In the next section, we first introduce a perfect schedule that is needed in the analysis in Secs. 6 and 7.

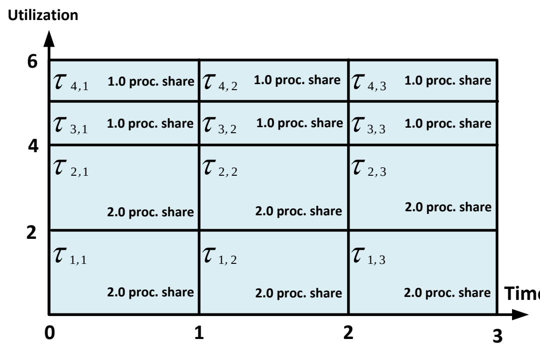


Fig. 4: PS schedule of the tasks in Example 1.

5 PS Schedule

To keep track of the workload completed in the GEDF-H schedule, we introduce the following PS schedule which is a perfect schedule in theory.

PS schedule. For any given sporadic task system τ, a *processor share* (PS) schedule is an ideal schedule where each task τ_i executes with a speed equal to u_i when it is pending (which ensures that each job of τ_i completes exactly at its deadline).

Fig. 4 shows the PS schedule of the tasks in Example 1. Note that the PS schedules does not depend on processor platform.

Our response time bound is obtained by comparing the workload done by jobs in the GEDF-H schedule S and the corresponding PS schedule, and quantifying the difference between the two. Let $A(\tau_{i,j}, t_1, t_2, S)$ and $A(\tau_{i,j}, t_1, t_2, PS)$ respectively denote the total workload done by $\tau_{i,j}$ in S and PS within $[t_1, t_2)$. Then, the total workload done by each task τ_i and all tasks in τ in $[t_1, t_2)$ under GEDF-H is given by

$$A(\tau_i, t_1, t_2, S) = \sum_{j \ge 1} A(\tau_{i,j}, t_1, t_2, S)$$

and

$$A(\tau, t_1, t_2, S) = \sum_{i=1}^{n} A(\tau_i, t_1, t_2, S).$$

$A(\tau_i, t_1, t_2, PS)$ and $A(\tau, t_1, t_2, PS)$ can be defined in a similar manner corresponding to PS schedule.

The difference between the workload done by a job $\tau_{i,j}$ up to time t in PS and S, denoted *the lag of job $\tau_{i,j}$ at time t in schedule S*, is defined by

$$lag(\tau_{i,j}, t, S) = A(\tau_{i,j}, 0, t, PS) - A(\tau_{i,j}, 0, t, S).$$

Similarly, the difference between the workload done by a task τ_i up to time t in PS and S, denoted *the lag of task τ_i at time t*, is defined by

$$\begin{aligned} & lag(\tau_i, t, S) \\ = & \sum_{j \ge 1} lag(\tau_{i,j}, t, S) \\ = & \sum_{j \ge 1} \big(A(\tau_{i,j}, 0, t, PS) - A(\tau_{i,j}, 0, t, S) \big). \end{aligned} \tag{3}$$

The LAG for a certain job set β at time t in schedule S is defined as

$$LAG(\beta, t, S) = \sum_{\tau_{i,j} \in \beta} lag(\tau_{i,j}, t, S). \quad (4)$$

The following property follows from the definitions above.

Property 3. Let β be a set of jobs. If $LAG(\beta, t_2, S) > LAG(\beta, t_1, S)$, where $t_2 > t_1$, then $[t_1, t_2)$ is non-busy for β.

Proof: We prove by contradiction. Suppose $[t_1, t_2)$ is busy for β. By this supposition,

$$\sum_{\tau_{i,j} \in \beta} A(\tau_{i,j}, t_1, t_2, S) = (t_2 - t_1) \cdot R_{sum} \quad (5)$$

and by the definition of PS schedule,

$$\sum_{\tau_{i,j} \in \beta} A(\tau_{i,j}, t_1, t_2, PS) = (t_2 - t_1) \cdot U_{sum}. \quad (6)$$

Thus, by the definition of LAG

$$\begin{aligned} & LAG(\beta, t_2, S) - LAG(\beta, t_1, S) \\ = & \sum_{\tau_{i,j} \in \beta} A(\tau_{i,j}, t_1, t_2, PS) - \sum_{\tau_{i,j} \in \beta} A(\tau_{i,j}, t_1, t_2, S) \\ = & (t_2 - t_1) \cdot U_{sum} - (t_2 - t_1) \cdot R_{sum} \\ \leq & 0, \end{aligned}$$

which is a contradiction.

□

6 Preemptive GEDF-H Analysis

We now present our analysis for preemptive GEDF-H scheduling. Our goal is to prove that the bounded response time can be guaranteed by preemptive GEDF-H without the loss of utilization.

Let τ_k be an arbitrary task in τ and $\tau_{k,l}$ be an arbitrary job of task τ_k, $t_d = d_{k,l}$, and S be a GEDF-H schedule for τ with the following inductive assumption.

Assumption 1. The response time of every job $\tau_{i,j}$, where $\tau_{i,j}$ has a higher priority than $\tau_{k,l}$, is at most $x + 2 \cdot T_i$ in S, where $x \geq 0$.

Our objective is to determine an x such that the response time of $\tau_{k,l}$ is at most $x + 2 \cdot T_k$ under this assumption. If we can find such x, by induction, this implies a response time of at most $x + 2 \cdot T_i$ for all jobs of every task τ_i, where $\tau_i \in \tau$.

Definition 6. We categorize jobs based on the relationship between their priorities and those of $\tau_{k,l}$:

$$\mathbf{H} = \{\tau_{i,j} : (d_{i,j} < t_d) \vee (d_{i,j} = t_d \wedge i \leq k)\}.$$

Thus, $\mathbf{H}$ is the set of jobs with priority no less than that of $\tau_{k,l}$.

By Def. 6, $\tau_{k,l}$ is in $\mathbf{H}$. Also jobs not in $\mathbf{H}$ have lower priorities than those in $\mathbf{H}$ and thus do not affect the scheduling of jobs in $\mathbf{H}$. For simplicity, in the rest of the analysis in this section, we get rid of the jobs that are not in $\mathbf{H}$ from the task system τ. Thus,

$$LAG(\mathbf{H}, t, S) = LAG(\tau, t, S) = \sum_{\tau_i \in \tau} lag(\tau_i, t, S) \quad (7)$$

Because each job in $\mathbf{H}$ completes before or at t_d in the PS schedule, $LAG(\mathbf{H}, t_d, S)$ denotes the total workload of jobs in $\mathbf{H}$ pending at t_d in the GEDF-H schedule (i.e, the workload that is able to compete against $\tau_{k,l}$ after t_d). Intuitively, the response time of $\tau_{k,l}$ cannot exceed $x+2 \cdot T_k$ if the value of $LAG(\mathbf{H}, t_d, S)$ is adequately small.

The steps for determining the value for x are as follows.

1) Determine a safe bound on $LAG(\mathbf{H}, t_d, S)$ required for the response time of $\tau_{k,l}$ to exceed $x + 2 \cdot T_k$, under the Assumption 1. This is dealt with in Lemma 1 in Sec. 6.1.
2) Determine an upper bound on $LAG(\mathbf{H}, t_d, S)$ under the Assumption 1. This is dealt with in Lemmas 2 and 3 in Sec. 6.2.
3) Determine the smallest x such that the upper bound is no more than the safe bound. This is dealt with in Theorem 1 in Sec. 6.3.

6.1 Safe Bound on $LAG(\mathbf{H}, t_d, S)$

Lemma 1 below provides a safe bound on $LAG(\mathbf{H}, t_d, S)$. The response time of $\tau_{k,l}$ does not exceed $x + 2 \cdot T_k$ when $LAG(\mathbf{H}, t_d, S)$ is lower than that safe bound and Assumption 1 holds.

Lemma 1. *If $LAG(\mathbf{H}, t_d, S) \leq R_{sum} \cdot x + T_k$ and Assumption 1 holds, then the response time of $\tau_{k,l}$ is at most $x + 2 \cdot T_k$,*

Proof: Let $\eta_{k,l}$ be the amount of work $\tau_{k,l}$ performs by time t_d in S, $0 \leq \eta_{k,l} < C_k$. Define y as follows.

$$y = x + \frac{\eta_{k,l}}{R_{sum}} \quad (8)$$

We consider two cases.

Case 1. *$[t_d, t_d + y)$ is a busy interval for* $\mathbf{H}$. In this case, the amount of work completed in $[t_d, t_d + y)$ is exactly $\sum_{i=1}^{p} \alpha_i \cdot M_i \cdot y = R_{sum} \cdot y$, as illustrated in Fig. 5. Hence, the amount of work pending at $t_d + y$ is at most $LAG(\mathbf{H}, t_d, S) - R_{sum} \cdot y \leq R_{sum} \cdot x + T_k - R_{sum} \cdot x - \eta_{k,l} = T_k - \eta_{i,j}$. This remaining work will be completed no later than $t_d + y + T_k - \eta_{k,l} = t_d + x + \frac{\eta_{k,l}}{R_{sum}} + T_k - \eta_{k,l} \leq t_d + x + T_k$, even on a slowest processor. Since that remaining work includes the work due for $\tau_{k,l}$, $\tau_{k,l}$ thus completes by $t_d + x + T_k$. The response time of $T_{k,l}$ is not more than $t_d + x + T_k - r_{k,l} = x + 2 \cdot T_k$.

Case 2. *$[t_d, t_d + y)$ is a non-busy interval for* $\mathbf{H}$. Let t_s be the earliest non-busy instant in $[t_d, t_d + y)$, as illustrated in Fig. 6. By Prop. 1, at most $m - 1$ tasks have pending jobs in $\mathbf{H}$ at t_s. Moreover, since no jobs

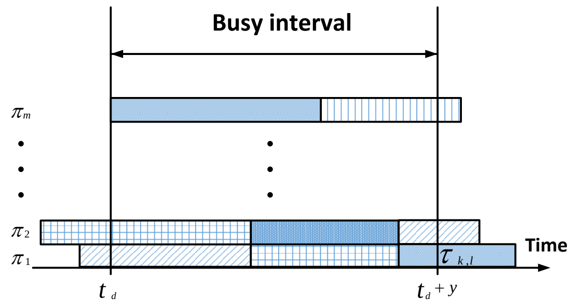


Fig. 5: $[t_d, t_d + y)$ is a busy interval.

in **H** can be released after t_d, we have the following property.

Property 4. At most $m - 1$ tasks have pending jobs in **H** at or after t_s. This implies no job in **H** would be preempted at or after t_s.

There are two sub-cases to consider.

Case 2.1: $\tau_{k,l}$ is executing at t_s. By Props. 4 and 2, we have

$$\begin{aligned}
f_{k,l} &\leq t_s + \frac{C_k - \eta_{k,l}}{v_{\pi_k}} \\
&\quad \{\text{by Eq. (2)}\} \\
&\leq t_d + y + \frac{C_k - \eta_{k,l}}{u_k} \\
&\quad \{\text{by Eq. (8)}\} \\
&\leq t_d + x + \frac{\eta_{k,l}}{R_{sum}} + \frac{C_i - \eta_{k,l}}{u_k} \\
&\leq t_d + x + \frac{C_k}{u_k} \\
&= t_d + x + T_k.
\end{aligned}$$

Thus, the response time of $\tau_{k,l}$ is not more than $f_{k,l} - r_{k,l} = f_{k,l} - t_d + T_k \leq x + 2 \cdot T_k$.

Case 2.2: $\tau_{k,l}$ is not executing at t_s. In this case, $\eta_{k,l} = 0$, and the predecessor job $\tau_{k,l-1}$ has not completed by t_s. Because $d_{k,l-1} = t_d - T_k < t_d$, by Assumption 1, $f_{k,l-1} \leq r_{k,l-1} + x + 2 \cdot T_k \leq r_{k,l} + x + T_k$. Thus, combined with Props. 4 and 2,

$$\begin{aligned}
f_{k,l} &\leq f_{k,k-1} + \frac{C_k}{v_{\pi_k}} \\
&\leq r_{k,l} + x + T_k + \frac{C_k}{v_{\pi_k}} \\
&\quad \{\text{by Eq. (2)}\} \\
&\leq r_{k,l} + x + T_k + \frac{C_k}{u_k} \\
&= r_{k,l} + x + 2 \cdot T_k.
\end{aligned}$$

The response time of $\tau_{k,l}$ is thus not more than $x+2\cdot T_k$.

By the above discussion, in either case, Lemma 1 holds.

□

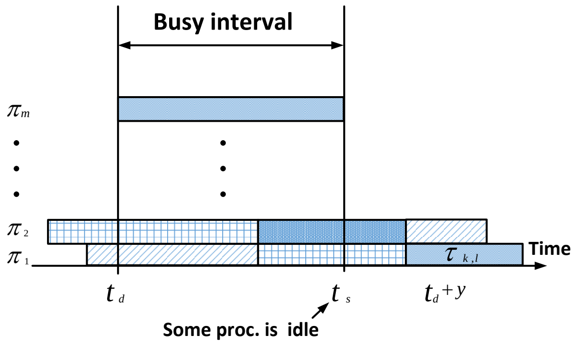


Fig. 6: $[t_d, t_d + y)$ is a non-busy interval.

6.2 Upper Bound on $LAG(\mathbf{H}, t_d, S)$

In this section, we determine an upper bound on $LAG(\mathbf{H}, t_d, S)$ under Assumption 1.

Definition 7. Let $t_n \leq t_d$ be the latest non-busy instant by t_d for **H**, if any; otherwise, $t_n = 0$.

By the above definition and Prop. 3, we have

$$LAG(\mathbf{H}, t_d, S) \leq LAG(\mathbf{H}, t_n, S). \tag{9}$$

Lemma 2. *For any task τ_i, if τ_i has one or more pending jobs at t_n in the preemptive GEDF-H schedule S, then we have*

$$lag(\tau_i, t_n, S) \leq \begin{cases} C_i & \text{if } d_{i,j} \geq t_n \\ u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}} & \text{if } d_{i,j} < t_n \end{cases}$$

where $d_{i,j}$ is the deadline of the earliest released pending job of τ_i, $\tau_{i,j}$, at time t_n in S.

Proof: Let $\eta_{i,j}$ ($\eta_{i,j} < C_i$) be the amount of work $\tau_{i,j}$ performs before t_n in the preemptive GEDF-H schedule.

In the both PS schedule and preemptive GEDF-H schedule S, because the job released before $\tau_{i,j}$ has completed, we have $lag(\tau_i, t_n, S) = \sum_{h \geq j} lag(\tau_{i,h}, t_n, S) = \sum_{h \geq j} \big(A(\tau_{i,h}, 0, t_n, PS) - A(\tau_{i,h}, 0, t_n, S)\big)$. By the definition of $A(\tau_{i,j}, t_1, t_2, S)$, $A(\tau_{i,h}, 0, t_n, S) = A(\tau_{i,h}, r_{i,h}, t_n, S)$. Thus,

$$\begin{aligned}
&lag(\tau_i, t_n, S) \\
= \; &A(\tau_{i,j}, r_{i,j}, t_n, PS) - A(\tau_{i,j}, r_{i,j}, t_n, S) \\
&+ \sum_{h>j} \big(A(\tau_{i,h}, r_{i,h}, t_n, PS) \\
&- A(\tau_{i,h}, r_{i,h}, t_n, S)\big).
\end{aligned} \tag{10}$$

By the definition of PS schedule,

$$A(\tau_{i,j}, r_{i,j}, t_n, PS) \leq C_i,$$

and

$$\sum_{h>j} A(\tau_{i,h}, r_{i,h}, t_n, PS) \leq u_i \cdot \max(0, t_n - d_{i,j}).$$

In the preemptive GEDF-H schedule S, by the selection of $\tau_{i,j}$,

$$A(\tau_{i,j}, r_{i,j}, t_n, S) = \eta_{i,j}$$

and

$$\sum_{h>j} A(\tau_{i,h}, r_{i,h}, t_n, S) = 0.$$

By setting these values into Eq. (10), we have

$$lag(\tau_i, t_n, S) \le C_i - \eta_{i,j} + u_i \cdot \max(0, t_n - d_{i,j}). \quad (11)$$

There are two cases to consider.

Case 1. $d_{i,j} \ge t_n$**.** In this case, Eq. (11) implies $lag(\tau_i, t_n, S) \le C_i - \eta_{i,j} \le C_i$.

Case 2. $d_{i,j} < t_n$**.** In this case, the earliest possible completion time of $\tau_{i,j}$ is $t_n + \frac{C_i - \eta_{i,j}}{\alpha_{max}}$ (executes on the fastest processor without being preempted). Thus,

$$f_{i,j} \ge t_n + \frac{C_i - \eta_{i,j}}{\alpha_{max}} \quad (12)$$

Because $d_{i,j} < t_n = t_d$, $\tau_{i,j}$ is not the job $\tau_{k,l}$. Thus, by Assumption 1, the response time of $\tau_{i,j}$ is at most $x + 2 \cdot T_i$. Hence,

$$f_{i,j} - r_{i,j} \le x + 2 \cdot T_i = x + T_i + d_{i,j} - r_{i,j} \quad (13)$$

By, Eqs. (12) and (13),

$$t_n - d_{i,j} \le x + \frac{\eta_{i,j}}{\alpha_{max}} + T_i - \frac{C_i}{\alpha_{max}}. \quad (14)$$

Setting this value into Eq. (11), we have

$$\begin{aligned} & lag(\tau_i, t_n, S) \\ \le \; & C_i - \eta_{i,j} + u_i \cdot (x + \frac{\eta_{i,j}}{\alpha_{max}} + T_i - \frac{C_i}{\alpha_{max}}) \\ \le \; & u_i \cdot x + C_i + u_i \cdot (T_i - \frac{C_i}{\alpha_{max}}) \\ = \; & u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}} \quad (15) \end{aligned}$$

Lemma 2 thus follows. □

As a consequence of the above discussion, Lemma 3 below upper bounds $LAG(\mathbf{H}, t_d, S)$.

Lemma 3. *With Assumption 1,* $LAG(\boldsymbol{H}, t_d, S) \le \overline{U}_{m-1} \cdot x + 2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}$.

Proof: The upper bound of $LAG(\mathbf{H}, t_n, S)$ can be derived by summing the individual task lags at t_n. Given that the instant t_n is non-busy, by Prop. 1, at most $m-1$ tasks have pending jobs at t_n. Let ξ denote the set of such tasks. Note that, if a task τ_i does not have a pending job at t_n in the preemptive GEDF-H schedule S, $lag(\tau_i, t_n, S) \le 0$. Therefore, by Eq. (9), we have

$$\begin{aligned} & LAG(\mathbf{H}, t_d, S) \\ \le \; & LAG(\mathbf{H}, t_n, S) \\ = \; & \sum_{\tau_i \in \tau} lag(\tau_i, t_n, S) \;\; \{\text{by Eq. (7)}\} \\ \le \; & \sum_{\tau_i \in \xi} lag(\tau_i, t_n, S) \\ & \{\text{by Lemma 2}\} \\ \le \; & \sum_{\tau_i \in \xi} \left(u_i \cdot x + C_i + u_i \cdot \left(T_i - \frac{C_i}{\alpha_{max}} \right) \right) \\ = \; & \sum_{\tau_i \in \xi} \left(u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}} \right) \\ = \; & \sum_{\tau_i \in \xi} u_i \cdot x + \sum_{\tau_i \in \xi} 2 \cdot C_i - \sum_{\tau_i \in \xi} \frac{u_i \cdot C_i}{\alpha_{max}} \quad (16) \end{aligned}$$

By Def. 1,

$$\begin{aligned} & \sum_{\tau_i \in \xi} u_i \cdot x + \sum_{\tau_i \in \xi} 2 \cdot C_i - \sum_{\tau_i \in \xi} \frac{u_i \cdot C_i}{\alpha_{max}} \\ \le \; & \overline{U}_{m-1} \cdot x + 2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} \end{aligned}$$

Lemma 3 thus follows. □

6.3 Determining the value of x

Setting the upper bound on $LAG(\mathbf{H}, t_d, S)$ in Lemma 3 to be at most the safe bound in Lemma 1 will ensure that the response time of $\tau_{k,l}$ is at most $x + 2 \cdot T_k$. The resulting inequality can be used to determine a value for x. By Lemmas 1 and 3, this inequality is $R_{sum} \cdot x + T_k \ge \overline{U}_{m-1} \cdot x + 2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}$. Solving for x, to make a x valid for all tasks, we have

$$x \ge \frac{2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}. \quad (17)$$

By $U_{sum} \le R_{sum}$ and Def. 1, $\overline{U}_{m-1} < R_{sum}$ clearly holds. Let

$$x^* = max(0, \frac{2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}), \quad (18)$$

then the response time of any job $\tau_{i,j}$ will not exceed $x^* + 2 \cdot T_i$ in S.

By the above discussion, the theorem below follows.

Theorem 1. *The response time of any task τ_i scheduled under preemptive GEDF-H is at most $x + 2 \cdot T_i$, provided $U_{sum} \le R_{sum}$, where*

$$x = max(0, \frac{2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}), \quad (19)$$

7 Non-preemptive GEDF-H Analysis

We now present our analysis for non-preemptive GEDF-H scheduling.

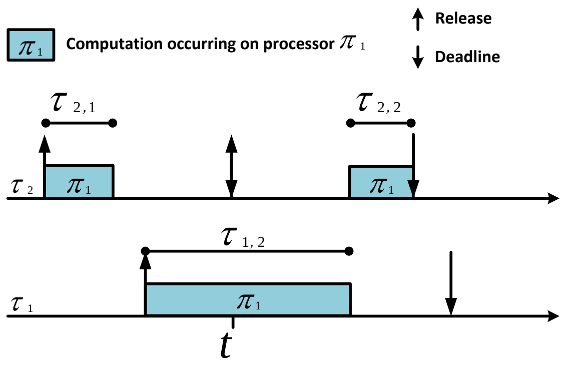


Fig. 7: Blocking instant.

Let τ_k be an arbitrary task in τ and $\tau_{k,l}$ be an arbitrary job of task τ_k, $t_d = d_{k,l}$, and S^* be a non-preemptive GEDF-H schedule S^* for τ with the following assumption.

Assumption 2. The response time of every job $\tau_{i,j}$, where $\tau_{i,j}$ has higher priority than $\tau_{k,l}$ is at most $x + 2 \cdot T_i$ in S^*, where $x \geq 0$.

In the analysis of GEDF-H scheduling, only the workload pending for jobs in $\mathbf{H}$ can compete with $\tau_{k,l}$. However, under non-preemptive GEDF-H, jobs not in $\mathbf{H}$ are still able to compete with $\tau_{k,l}$, because a job cannot be preempted once it starts to execute. Hence, the pending workload from blocking jobs should be taken into consideration. We first define following terms for the non-preemptive case.

Definition 8. Let $\mathbf{H}^*$ denote the set of jobs not in $\mathbf{H}$ that block one or more jobs in $\mathbf{H}$ at time instant t_d and may continue to execute at t_d under non-preemptive GEDF-H schedule S^*. Let $B(\mathbf{H}^*, t_d, S^*)$ denote the total workload pending for jobs in $\mathbf{H}^*$ at t_d.

Definition 9. For any time instant t, if there exists an $\varepsilon > 0$ such that during interval $[t, t+\varepsilon)$ there is an enabled job τ_{i_1,j_1} in $\mathbf{H}$ is not executing while any job τ_{i_2,j_2} in $\mathbf{H}^*$ is executing on some processor during this interval, we say τ_{i_1,j_1} is blocked by τ_{i_2,j_2} at time t where τ_{i_1,j_1} is a *blocked job*, τ_{i_2,j_2} is a *blocking job* and t is a *blocking instant*.

Fig. 7 illustrates the blocking instant. Suppose we have two tasks running on one processor. As shown in the figure, at time instant t, even though $\tau_{2,2}$ is enabled and has a higher priority than that of $\tau_{1,2}$, $\tau_{2,2}$ cannot execute under the non-preemptive GEDF-H. In this example, t is a blocking instant, $\tau_{1,2}$ is a blocking job and $\tau_{2,2}$ is a blocked job.

Definition 10. An interval $[a, b)$ is a *blocking interval* if every instant in it is a blocking instant. A blocking interval is said to be a *maximal blocking interval* if for any $c < a$, $[c, b)$ cannot be a blocking interval.

The following properties directly follows from the definition above.

Property 5. No processor is idle at any instant t within a blocking interval $[a, b)$.

Property 6. Consider a blocking interval $[a, b)$. Suppose job τ_{i_2,j_2} in $\mathbf{H}^*$ blocks one or more jobs in $\mathbf{H}$ at time instant b. Then τ_{i_2,j_2} must execute at time instant a.

Proof: Suppose t' is a blocking instant in $[a, b)$ and job τ_{i_1,j_1} in $\mathbf{H}$ is a blocked job at t'. Hence, GEDF-H scheduler would choose τ_{i_1,j_1} to execute at t' instead of τ_{i_2,j_2}. Thus τ_{i_2,j_2} cannot commence to execute at any time instant within $[a, b)$ and therefore must be executing at a. □

Property 7. Consider a maximal blocking interval $[a, b)$. If task τ_{i_1} is not executing at a, then it has no job in $\mathbf{H}$ pending at time instant a.

Proof: Let τ_{i_2,j_2} be a blocking job at time instant a. We prove by contradiction. Suppose τ_{i_1} is not executing at a and it has a job, namely τ_{i_1,j_1}, in $\mathbf{H}$ pending at a. Let r_{i_1,j_1} and s_{i_2,j_2} be the release time of τ_{i_1,j_1} and the start time of τ_{i_2,j_2}. By Def. 3, τ_{i_1,j_1} is released before a, i.e., $r_{i_1,j_1} < a$; by and Prop. 6, $s_{i_2,j_2} < a$. Thus, τ_{i_1,j_2} is blocked by τ_{i_2,j_2} in $[max(r_{i_1,j_1}, s_{i_2,j_2}), a)$. This contradicts that $[a, b)$ is a maximal blocking interval.

□

Now let us consider how to find the response time bound for the non-preemptive case. By the discussion above, the total pending work that can compete with $\tau_{k,l}$ after t_d is given by

$$LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*). \tag{20}$$

Analogous to the analysis of preemptive GEDF-H schedule in Sec. 6, we follow the three steps below to determine the value of x for the non-preemptive case.

1) Determine a safe bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$ required for the response time of $\tau_{k,l}$ to exceed $x + 2 \cdot T_k$, under the Assumption 2. This is dealt with in Lemma 4 in Sec. 7.1.
2) Determine an upper bound on $LAG(\mathbf{H}, t_d, S) + B(\mathbf{H}^*, t_d, S^*)$ under the Assumption 2. This is dealt with in Lemma 7 in Sec. 7.2.
3) Determine the smallest x such that the response time of $\tau_{k,l}$ is at most $x + 2 \cdot T_k$, using the above safe and upper bounds. This is dealt with in Theorem 2 in Sec. 7.3.

7.1 Safe Bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$

To find a safe bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$, we have the following parallel lemma for non-preemptive GEDF-H schedule S^*.

Lemma 4. *If $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*) \leq R_{sum} \cdot x + T_k$ and the Assumption 2 holds, then the response time of $\tau_{k,l}$ is at most $x + 2 \cdot T_k$.*

Proof: This proof is similar to the proof of Lemma 1. We only show the sketch of the proof. Let $\eta_{k,l}$ be the amount of work $\tau_{k,l}$ performs by time t_d in S, $0 \leq \eta_{k,l} < C_k$. Define y as follows.

$$y = x + \frac{\eta_{k,l}}{R_{sum}} \tag{21}$$

We consider the following two cases.

1) **Case 1.** $[t_d, t_d+y)$ **is a busy interval for** $\mathbf{H}\bigcup\mathbf{H}^*$.
2) **Case 2.** $[t_d, t_d+y)$ **is a non-busy interval for** $\mathbf{H}\bigcup\mathbf{H}^*$.

Following the same analysis in the proof of Lemma 1, Lemma 4 directly follows. □

7.2 Upper Bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$

For the non-preemptive case, in order to find the upper bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$, we first present the analysis of the maximal blocking interval.

Lemma 5. *Suppose $[t_1, t_2)$, where $t_2 \leq t_d$ is a maximal blocking interval. If τ_i has a job in $\boldsymbol{H}$ that is executing at t_1, then*

$$lag(\tau_i, t_1, S^*) \leq u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}}. \tag{22}$$

Proof: Let $\tau_{i,j}$ be the job of τ_i in $\mathbf{H}$ that is executing at t_1 and $\eta_{i,j}$ ($\eta_{i,j} < C_i$) be the amount of work $\tau_{i,j}$ performs before t_1 in the non-preemptive GEDF-H schedule S^*.

Case 1: $d_{i,j} < t_1$**.** Because $\tau_{i,j}$ cannot be preempted, the earliest time instant at which $\tau_{i,j}$ could complete is $t_1 + \frac{C_i - \eta_{i,j}}{\alpha_{max}}$. We have

$$f_{i,j} \geq t_1 + \frac{C_i - \eta_{i,j}}{\alpha_{max}}. \tag{23}$$

and, by Assumption 2,

$$f_{i,j} - r_{i,j} \leq x + 2 \cdot T_i. \tag{24}$$

Thus,

$$\begin{aligned}
t_1 - d_{i,j} &= t_1 - (r_{i,j} + T_i)\\
&\quad \{by\ Eq.\ (24)\}\\
&\leq t_1 - (f_{i,j} - x - T_i)\\
&\quad \{by\ Eq.\ (23)\}\\
&\leq \frac{\eta_{i,j} - C_i}{\alpha_{max}} + x + T_i
\end{aligned} \tag{25}$$

Because both in the PS schedule and non-preemptive GEDF-H schedule the job of τ_i released before $\tau_{i,j}$ has completed by t_1, $lag(\tau_i, t_1, S^*) = \sum_{h \geq j} lag(\tau_{i,h}, t_1, S^*)$. In the PS schedule,

$$\sum_{h \geq j} A(\tau_{i,h}, 0, t_1, PS) = (t_1 - r_{i,j}) \cdot u_i \tag{26}$$

and in the non-preemptive GEDF-H schedule,

$$\sum_{h \geq j} A(\tau_{i,h}, 0, t_1, S^*) = \eta_{i,j} \tag{27}$$

Thus,

$$\begin{aligned}
& lag(\tau_i, t_1, S^*)\\
=\ & \sum_{h \geq j} lag(\tau_{i,h}, t_1, S^*)\\
& \{by\ Eq.\ (3)\}\\
=\ & \sum_{h \geq j} A(\tau_{i,h}, 0, t_1, PS) - \sum_{h \geq j} A(\tau_{i,h}, 0, t_1, S^*)\\
& \{by\ Eqs.\ (26)\ and\ (27)\}\\
=\ & (t_1 - r_{i,j}) \cdot u_i - \eta_{i,j}\\
=\ & (t_1 - d_{i,j} + T_i) \cdot u_i - \eta_{i,j}\\
=\ & (t_1 - d_{i,j}) \cdot u_i + C_i - \eta_{i,j}\\
& \{by\ Eq.\ (25)\}\\
\leq\ & (\frac{\eta_{i,j} - C_i}{\alpha_{max}} + x + T_i) \cdot u_i + C_i - \eta_{i,j}\\
=\ & u_i \cdot x + C_i + u_i \cdot (T_i - \frac{C_i}{\alpha_{max}}) + (\frac{u_i}{\alpha_{max}} - 1) \cdot \eta_{i,j}\\
\leq\ & u_i \cdot x + C_i + u_i \cdot (T_i - \frac{C_i}{\alpha_{max}})\\
=\ & u_i \cdot x + 2 \cdot C_i + - \frac{u_i \cdot C_i}{\alpha_{max}}
\end{aligned}$$

Case 2: $d_{i,j} \geq t_1$**.** In this case, $t_1 - r_{i,j} \leq T_i$ and Eqs. 26 and 26 still hold. Thus, similar to Case 1,

$$\begin{aligned}
& lag(\tau_i, t_1, S^*)\\
=\ & \sum_{h \geq j} lag(\tau_{i,h}, t_1, S^*)\\
=\ & \sum_{h \geq j} A(\tau_{i,h}, 0, t_1, PS) - \sum_{h \geq j} A(\tau_{i,h}, 0, t_1, S^*)\\
=\ & (t_1 - r_{i,j}) \cdot u_i - \eta_{i,j}\\
\leq\ & T_i \cdot u_i - \eta_{i,j}\\
=\ & C_i - \eta_{i,j}\\
\leq\ & C_i
\end{aligned} \tag{28}$$

Therefore, in either case, Lemma 5 holds. □

The following lemma is crucial for the analysis of non-preemptive GEDF-H.

Lemma 6. *Suppose $[t_1, t_2)$ is a blocking interval. Then, $LAG(\boldsymbol{H}, t_2, S^*) + B(\boldsymbol{H}^*, t_2, S^*) < LAG(\boldsymbol{H}, t_1, S^*) + B(\boldsymbol{H}^*, t_1, S^*)$.*

Proof: Let W_d and W_{d^*} respectively be the amount of workload done by jobs in $\mathbf{H}$ and $\mathbf{H}^*$ during $[t_1, t_2)$ in the non-preemptive GEDF-H schedule S^*. Therefore,

$$A(\tau, t_1, t_2, S^*) = W_d$$

and, in the PS schedule,

$$A(\tau, t_1, t_2, PS) = (t_2 - t_1) \cdot U_{sum}.$$

Therefore,

$$\begin{aligned}&LAG(\mathbf{H}, t_2, S^*) - LAG(\mathbf{H}, t_1, S^*)\\ =\ & A(\tau, t_1, t_2, PS) - A(\tau, t_1, t_2, S^*)\\ =\ & (t_2 - t_1) \cdot U_{sum} - W_d \qquad (29)\end{aligned}$$

By the definition of $B(\mathbf{H}^*, t, S^*)$,

$$B(\mathbf{H}^*, t_2, S^*) = B(\mathbf{H}^*, t_1, S^*) - W_{d^*} \qquad (30)$$

By Prop. 5, no processor is idle at any time instant within $[t_1, t_2)$, thus $W_d + W_{d^*} = (t_2 - t_1) \cdot R_{sum}$. As a consequence of the above discussion,

$$\begin{aligned}&LAG(\mathbf{H}, t_2, S^*) - LAG(\mathbf{H}, t_1, S^*) + B(\mathbf{H}^*, t_2, S^*)\\ &\{by\ Eqs.\ (29)\ and\ (30)\}\\ =\ & B(\mathbf{H}^*, t_1, S^*) + (t_2 - t_1) \cdot U_{sum} - W_d - W_{d^*}\\ =\ & B(\mathbf{H}^*, t_1, S^*) + (t_2 - t_1) \cdot (U_{sum} - R_{sum})\\ \leq\ & B(\mathbf{H}^*, t_1, S^*) \qquad (31)\end{aligned}$$

Lemma 6 thus follows. □

After the above preparing work, we try to find an upper bound on $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$.

Lemma 7. *With Assumption 2, $LAG(\boldsymbol{H}, t_d, S^*) + B(\boldsymbol{H}^*, t_d, S^*) \leq \overline{U}_{m-1} \cdot x + \overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}$.*

Proof: we consider two cases depending on whether t_d is a blocking instant.

Case 1: t_d is not a blocking instant. In this case, by Def. 9, $\mathbf{H}^*$ is empty and $B(\mathbf{H}^*, t_d, S^*) = 0$. Thus, the analysis of this case is the same as that of the preemptive case as shown in Lemmas 2 and 3. We have

$$\begin{aligned}&LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)\\ =\ & LAG(\mathbf{H}, t_d, S^*)\\ \leq\ & \overline{U}_{m-1} \cdot x + 2 \cdot \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}\\ \leq\ & \overline{U}_{m-1} \cdot x + \overline{C}_{m-1} + \overline{C}_m - \frac{\overline{V}_{m-1}}{\alpha_{max}}\end{aligned}$$

Case 2: t_d is a blocking instant. Let $[t^*, t_d)$ be the maximal blocking interval.

We first consider the upper bound on $LAG(\mathbf{H}, t^*, S^*) + B(\mathbf{H}^*, t^*, S^*)$. By Prop. 7, if task τ_i is not executing at t^*, it has no job in $\mathbf{H}$ pending at t^* and, by Prop 6, it has not job in $\mathbf{H}^*$. Thus, such tasks cannot contribute to either $LAG(\mathbf{H}, t^*, S^*)$ or $B(\mathbf{H}^*, t^*, S^*)$. Let ξ_d be set the of tasks that have a job in $\mathbf{H}$ that is executing at t^* and ξ_{d^*} be set the of tasks that have a job in $\mathbf{H}^*$ that is executing at t^*. Thus,

$$\begin{aligned}&LAG(\mathbf{H}, t^*, S^*)\\ =\ & \sum_{\tau_i \in \xi_d} lag(\tau_i, t^*, S^*)\\ &\{by\ Lemma\ (5)\}\\ \leq\ & \sum_{\tau_i \in \xi_d} \left(u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}}\right) \qquad (32)\end{aligned}$$

By Prop. 6, each task τ_i in ξ_{d^*} has only one job in $\mathbf{H}^*$ so it contributes at most C_i to $B(\mathbf{H}^*, t^*, S^*)$. Thus,

$$B(\mathbf{H}^*, t^*, S^*) \leq \sum_{\tau_i \in \xi_{d^*}} C_i \qquad (33)$$

Note that there are m tasks in $\xi_d \bigcup \xi_{d^*}$ and at most $m-1$ tasks in ξ_d. By Eqs. (32) and (33), we have

$$\begin{aligned}&LAG(\mathbf{H}, t^*, S^*) + B(\mathbf{H}^*, t^*, S^*)\\ \leq\ & \sum_{\tau_i \in \xi_d} \left(u_i \cdot x + 2 \cdot C_i - \frac{u_i \cdot C_i}{\alpha_{max}}\right) + \sum_{\tau_i \in \xi_{d^*}} C_i\\ =\ & \sum_{\tau_i \in \xi_d} \left(u_i \cdot x + C_i + \frac{u_i \cdot C_i}{\alpha_{max}}\right) + \sum_{\tau_i \in \xi_d \cup \xi_{d^*}} C_i\\ \leq\ & \overline{U}_{m-1} \cdot x + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} + \overline{C}_m \qquad (34)\end{aligned}$$

Combine with Lemma 6,

$$\begin{aligned}&LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)\\ \leq\ & LAG(\mathbf{H}, t^*, S^*) + B(\mathbf{H}^*, t^*, S^*)\\ \leq\ & \overline{U}_{m-1} \cdot x + \overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}\end{aligned}$$

Lemma 7 thus follows. □

7.3 Determining the values of x

Similar to the analysis in Sec. 6.3, in order to find a value for x that ensures the value of $LAG(\mathbf{H}, t_d, S^*) + B(\mathbf{H}^*, t_d, S^*)$ is less than the safe bound, we set the upper bound to be at most the safe bound. By Lemmas 4 and 7, this inequality is $R_{sum} \cdot x + T_k \geq \overline{U}_{m-1} \cdot x + \overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}}$. Solving for x, to make a x valid for all tasks, we have

$$x \geq \frac{\overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}.$$

By $U_{sum} \leq R_{sum}$ and Def. 1, $\overline{U}_{m-1} < R_{sum}$ clearly holds. Let

$$x^* = max\left(0, \frac{\overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}\right), \qquad (35)$$

then the response time of each job in τ_i will not exceed $x^* + 2 \cdot T_i$ in the non-preemptive GEDF-H schedule.

By the above discussion, the theorem below follows for the non-preemptive case.

Theorem 2. *The response time of any task τ_i scheduled under non-preemptive GEDF-H is at most $x + 2 \cdot T_i$,* *provided $U_{sum} \leq R_{sum}$, where*

$$x = max(0, \frac{\overline{C}_m + \overline{C}_{m-1} - \frac{\overline{V}_{m-1}}{\alpha_{max}} - T_{min}}{R_{sum} - \overline{U}_{m-1}}), \quad (36)$$

8 Experiment

Although GEDF-H ensures SRT schedulability with no utilization loss under both preemptive and non-preemptive scheduling, the magnitude of the resulting response time bound is also important. In this section, we describe the experiments conducted using randomly-generated task sets to evaluate the applicability of the response time bound given in Theorems 1, and 2. Our goal is to examine how large the magnitude of response time is.

8.1 Experimental setup

We set the processor platform in our experiment close to the Intel's QuickIA uniform heterogeneous prototype platform [7]. The QuickIA platform contains two kinds of processors and each kind contains two processors. We assume that two of the processors π_1 and π_2 have unit speed and the other two processors π_3 and π_4 have two-unit speed, i.e., $\alpha_1 = 1$ and $\alpha_2 = 2$. The unit time is assumed to be $1ms$.

By the definitions of Ψ and Φ, we have $\Psi_0 = \{\pi_1, \pi_2, \pi_3, \pi_4\}$, $|\Psi_0| = 4$, $\Psi_1 = \{\pi_3, \pi_4\}$ and $|\Psi_1| = 2$. We generated tasks as follows.

The minimum inter-arrival time T_i of tasks ranged from 100ms to 1000ms. First, we generated tasks in Φ_1. According to Eq. (1), $|\Phi_1| \leq |\psi_1| = 2$ and the utilization of tasks in Φ_1 is at most 2. We thus first randomly generated the number of tasks in Φ_1 from 0 to 2, and task utilizations were generated using the uniform distribution $(1, 2]$. Task execution costs were calculated from T_i and utilizations. Then, we generated tasks in Φ_0/Φ_1. The utilization of tasks in Φ_0/Φ_1 is not more than 1. In the first experiment, these task utilizations were generated using three uniform distributions:

1) Light:[0.001, 0.05]
2) Medium:[0.05, 0.2]
3) Heavy: [0.2, 0.5]

In the second experiment, the minimum inter-arrival time of each task was fixed by 100ms, 500ms and 1000ms, and the per-task utilization ranged from 0.1 to 1.

For each experiment, 100,000 task systems were generated. Each such task system was generated by creating tasks until total utilization exceeded $R_{sum} = 6$, and by then reducing the last task's utilization so that the total utilization equaled R_{sum}. We recorded the maximum response time bound, the minimum response time bound, and the average response time bound among the task systems.

8.2 Results Analysis

The obtained results are shown in Fig. 8 (the organization of which is explained in the figure's caption). Each graph in Fig. 8 contains six curses, which plots the calculated maximum response time bound, average response time bound, and minimum response time bound among all task systems in the system by using the formulas in Theorems 1, and 2, respectively.

In all of the six graphs, the differences of the response time bound between the two cases (i.e., preemptive and non-preemptive) are not significant, especially when the utilization is light as shown in Figs. 8(e).

In the first experiment, as seen in Figs. 8(a), (c), and (e), in all tested scenarios, the maximum response time bound is smaller than seven task relative deadlines, while the average response time bound is close to three task relative deadlines. One observation herein is that when task utilizations become heavier, the response time bounds increase. This is intuitive because the denominator of Eqs. (19) and (36) becomes smaller when task utilizations are heavier.

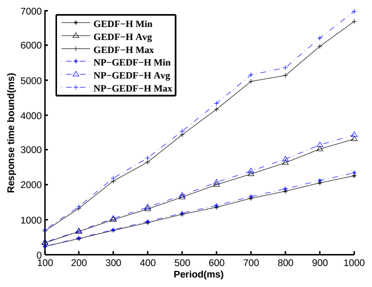
In the second experiment, under three fixed minimum inter-arrival time scenarios as seen in Figs. 8(b), (d), and (f), the response time bounds under GEDF-H slightly increase along with the increase of the average task utilization of the system. Specifically, we recorded the ratio of the derived response time bound to the relative deadline of each task. Fig. 9 clearly shows that such ratio is not larger 4 and smaller than 3 in the most cases.

Note that if we increase the maximum speed (i.e., α_{max}) of the processors without changing the total resource capacity (i.e., R_{sum}), the response time bound shall correspondingly increase, as suggested by Eqs. (19) and (36).

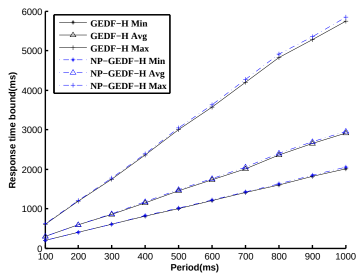
9 Simulation

In order to further evaluate the proposed scheduling algorithm and the corresponding response time bound, we have done simulations on randomly generated task systems scheduled under both the preemptive GEDF-H in Algorithm 1 and the original GEDF with random processor selection policy (termed as GEDF-R). The partitioning and semi-partitioning algorithms are not considered in our simulations, because in this paper we only focus on the global approach where the bin-packing-related loss can be avoided. We have done sets of simulations on task systems with different parameters and we could always have the following conclusions:

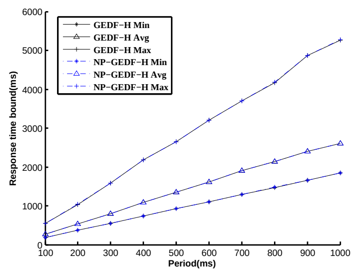
1) The response time bound derived in Theorem 1 is indeed a safe bound.
2) The theoretical response time bound is less than twice of the simulated response time.
3) GEDF-H is superior to GEDF-R with respect to the response time. Moreover, with the random



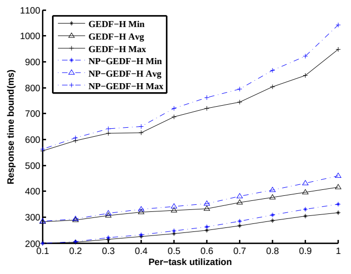
(a) Heavy task utilization



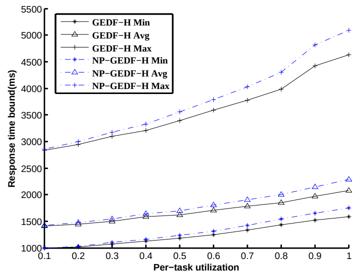
(b) Medium task utilization



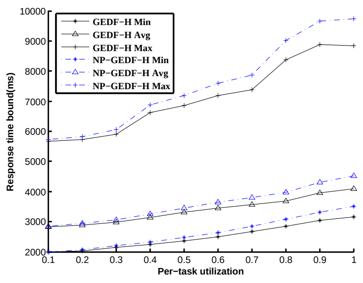
(c) Light task utilization



(d) Relative deadline = 100ms



(e) Relative deadline = 500ms



(f) Relative deadline = 1000ms

Fig. 8: Response time bounds. In all six graphs, the y-axis denotes the response time bound value. Each graph gives six curves plotting the maximum, average, and minimum response time bound among the generated task systems. The labels "**GEDF-H**" and "**NP-GEDF-H**" are used for the preemptive GEDF-H and non-preemptive GEDF-H, respectively. In the first column of graphs, the x-axis denotes the task periods. Heavy, medium, and light task utilizations are assumed in insets (a), (c), and (e), respectively. In the second column of graphs, the x-axis denotes the per-task utilization of the generated task system. Three specific values of minimum inter-arrival time, 100ms, 500ms, and 1000ms, are assumed in insets (b), (d), and (f), respectively.

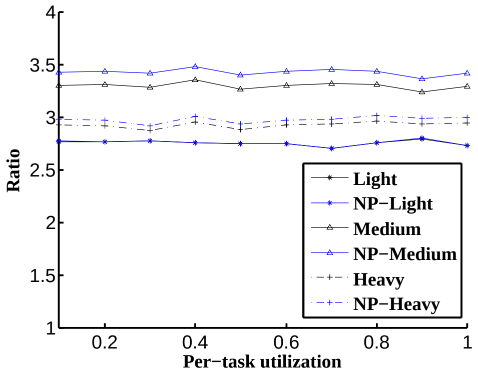


Fig. 9: Ratio of the derived response time bound to relative dealine

processor selection rule, the response time of the task is prone to grow to be unbounded.

Consider that the results of simulations with difference parameters are similar, we only show one set of simulation with detailed description.

9.1 Setup

We simulated six tasks running on two processors, where $\tau_1 = (60, 50)$, $\tau_2 = (20, 60)$, $\tau_3 = (40, 70)$, $\tau_4 = (20, 40)$, $\tau_5 = (20, 80)$, $\tau_6 = (10, 80)$, $v_{\pi_1} = 2$ and $v_{\pi_2} = 1$. In this set of simulation, we have $U_{sum} = 3 = R_{sum}$, which means the total utilization has reached its limitation. We simulated for 10000 time units and recorded the results for the first 500 jobs of each task.

9.2 Results Analysis

The simulation results are shown in Fig. 10 where the organization is explained in the figure's caption.

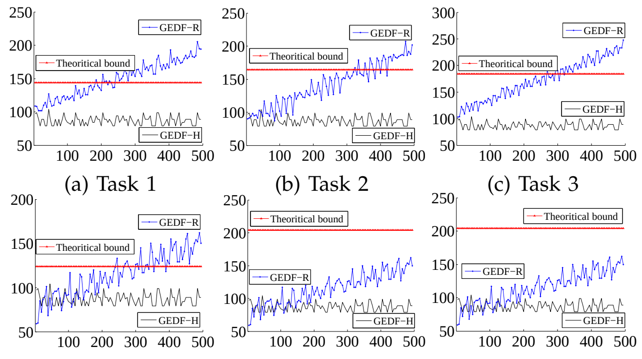
In this set of simulation, the response time of each job scheduled under preemptive GEDF-H never exceeds its theoretical bound, which verifies the correctness of our analysis in Sec. 6.

As shown in the figure, the response time bound derived in this paper is not larger than twice of the simulated response time. Given that the response time bound is derived by considering the worst-case scenarios, the difference between our bound and simulated response time is reasonable.

Moreover, the GEDF-H algorithm proposed in this paper is superior to the GEDF-R which randomly selects processors for tasks. As the figures suggest, the response time under GEDF-H fluctuates within a bounded range while it unboundedly increases along with the job ID under GEDF-R. Such a result implies that on a uniform heterogeneous multiprocessor platform, an appropriate processor selection strategy is critical.

10 Conclusion

We have shown that SRT sporadic task systems can be supported on a uniform heterogeneous multiprocessor with no utilization loss provided bounded response time is acceptable. The scheduling algorithm proposed in this paper is identical to GEDF except that it enforces a specific processor selection rule



(a) Task 1 (b) Task 2 (c) Task 3

(d) Task 4 (e) Task 5 (f) Task 6

Fig. 10: Simulation Results. In all six graphs, the y-axis and x-axis denote the response time (ms) and the job ID, respectively. In each graph, the horizontal line denotes the response time bound of each job scheduled under GEDF-H, and the lines labled "GEDF-H" and "GEDF-R" respectively show the response time of each job scheduled under GEDF-H and GEDF-R in the simulation.

which is critical on heterogeneous multiprocessors. GEDF-H and NP-GEDF-H could be implemented in real-time operating systems and hypervisors to coordinate processes or virtual machines with timing requirements. As demonstrated by experiments presented herein, GEDF-H is able to guarantee schedulability with no utilization loss while providing predicted response time. Besides, compared with the simulation results the response time bound derived in this paper is not only safe but also reasonably tight. Compared to GEDF, GEDF-H may incur more job migrations among processors due to the specific processor selection rule. However, without such a processor selection rule, the bounded response time may not be guaranteed. For the future work, we plan to design better algorithm that can reduce the job migration cost. Also it would be interesting to extent this work to hard-real systems and self-suspending task systems.

Acknowledgment

This work was supported by a start-up grant from the University of Texas at Dallas.

References

[1] Björn Andersson, Gurulingesh Raravi, and Konstantinos Bletsas. Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors. In *Proc. of the 31st Real-Time Systems Symposium*, pages 239–248. IEEE, 2010.

[2] Sanjoy Baruah. Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms. In *Proc. of the 25th Real-Time Systems Symposium*, pages 37–46. IEEE, 2004.

[3] Sanjoy Baruah and Nathan Fisher. The partitioned multiprocessor scheduling of sporadic task systems. In *Proc. of the 26th Real-Time Systems Symposium*, pages 9–pp. IEEE, 2005.

[4] Francois Bodin and Stephane Bihan. Heterogeneous multicore parallel programming for graphics processing units. *Scientific Programming*, 17(4):325–336, 2009.

[5] Jian-Jia Chen, Heng-Ruey Hsu, Kai-Hsiang Chuang, Chia-Lin Yang, Ai-Chun Pang, and Tei-Wei Kuo. Multiprocessor energy-efficient scheduling with task migration considerations. In *Proc. of the 16th Euromicro Conference on Real-Time Systems*, pages 101–108. IEEE, 2004.

[6] Jian-Jia Chen and Chin-Fu Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *RTCSA*, pages 28–38, 2007.

[7] Nagabhushan Chitlur, Ganapati Srinivasa, Scott Hahn, PK Gupta, Dheeraj Reddy, David A Koufaty, Paul Brett, Abirami Prabhakaran, Li Zhao, Nelson Ijih, et al. Quickia: Exploring heterogeneous architectures on real prototypes. In *Proc. of HPCA*, volume 12, pages 1–8, 2012.

[8] Jason Cong and Bo Yuan. Energy-efficient scheduling on heterogeneous multi-core architectures. In *Proc. of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 345–350. ACM, 2012.

[9] Umamaheswari C Devi. *Soft real-time scheduling on multiprocessors*. PhD thesis, University of North Carolina at Chapel Hill, 2006.

[10] Kenji Funaoka, Shinpei Kato, and Nobuyuki Yamasaki. Energy-efficient optimal real-time scheduling on multiprocessors. In *Proc. of the 11th International Symposium on Object Oriented Real-Time Distributed Computing*, pages 23–30. IEEE, 2008.

[11] Shelby Funk and Sanjoy Baruah. Task assignment on uniform heterogeneous multiprocessors. In *Proc. of the 17th Euromicro Conference on Real-Time Systems*, pages 219–226. IEEE, 2005.

[12] Shelby Funk, Joel Goossens, and Sanjoy Baruah. On-line scheduling on uniform multiprocessors. In *Proc. of the 22nd Real-Time Systems Symposium*, pages 183–192. IEEE, 2001.

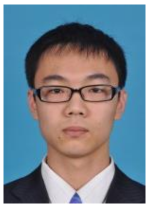
[13] Markus Happe, Enno Lübbers, and Marco Platzner. A self-adaptive heterogeneous multi-core architecture for embedded real-time video object tracking. *Journal of real-time image processing*, 8(1):95–110, 2013.

[14] Peter Kollig, Colin Osborne, and Tomas Henriksson. Heterogeneous multi-core platform for consumer multimedia applications. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1254–1259. European Design and Automation Association, 2009.

[15] Hennadiy Leontyev and James H Anderson. Tardiness bounds for edf scheduling on multi-speed multicore platforms. In *Proc. of the 13th Embedded and Real-Time Computing Systems and Applications*, pages 103–110. IEEE, 2007.

[16] Cong Liu, Jian Li, Wei Huang, Juan Rubio, Evan Speight, and Xiaozhu Lin. Power-efficient time-sensitive mapping in heterogeneous systems. In *Proc. of the 21st international conference on Parallel architectures and compilation techniques*, pages 23–32. ACM, 2012.

[17] Kecheng Yang and James H Anderson. Soft real-time semi-partitioned scheduling with restricted migrations on uniform heterogeneous multiprocessors. In *Proc. of the 22nd Real-Time Networks and Systems*, 2014.



Guangmo Tong is a Ph.D candidate in the Department of Computer Science at the University of Texas at Dallas. He received his BS degree in Mathematics and Applied Mathematics from Beijing Institute of Technology in July 2013. His research interests include real-time and embedded systems and social networks. He is a student member of the IEEE.

Cong Liu is an assistant professor in the Department of Computer Science at the University of Texas at Dallas. He received his Ph.D. degree in Computer Science from the University of North Carolina at Chapel Hill in July 2013. His research interests include real-time and embedded systems, operating systems, cloud computing, and wireless sensor networks. He has published over 30 papers in premier conferences and journals such as RTSS, PACT, ECRTS, RTAS, and JPDC. He received the Best Student Paper Award at the 30th IEEE Real-Time Systems Symposium, the premier real-time and embedded systems conference. He also received the best papers award at the 17th RTCSA and the UNC's dissertation fellowship.