A Stochastic Framework for Multiprocessor Soft Real-Time Scheduling*

Alex F. Mills
Department of Statistics and Operations Research
University of North Carolina at Chapel Hill

James H. Anderson
Department of Computer Science
University of North Carolina at Chapel Hill

Abstract

Prior work has shown that the global earliest-deadline-first (GEDF) scheduling algorithm ensures bounded deadline tardiness on multiprocessors with no utilization loss; therefore, GEDF may be a good candidate scheduling algorithm for soft real-time workloads. However, such workloads are often implemented assuming an average-case provisioning, and in prior tardiness-bound derivations for GEDF, worst-case execution costs are assumed. As worst-case costs can be orders of magnitude higher than average-case costs, using a worst-case provisioning may result in significant wasted processing capacity. In this paper, prior tardiness-bound derivations for GEDF are generalized so that execution times are probabilistic, and a bound on expected (mean) tardiness is derived. It is shown that, as long as the total expected utilization is strictly less than the number of available processors, the expected tardiness of every task is bounded under GEDF. The result also implies that any quantile of the tardiness distribution is also bounded.

1 Introduction

The advent of multicore platforms has led to renewed interest in multiprocessor scheduling techniques for real-time workloads. In this paper, we consider an important category of such workloads, namely, those with *soft* timing constraints. The specific such constraint that we consider is that *deadline tardiness* be bounded. We consider this constraint in the context of implicit-deadline sporadic task systems.

In work on such systems, Leontyev and Anderson showed that a variety of global scheduling algorithms are capable of ensuring bounded tardiness without utilization loss [4]. This result extended an earlier proof by Devi and Anderson that showed the same of the *global earliest-deadline-first* (GEDF) scheduling algorithm [1]. Partly because of this result, GEDF is currently being seriously considered for support in real-time Linux [9] (as defined by the REALTIME PREEMPT patch).

While these tardiness-bound results bode favorably for the viability of certain global algorithms like GEDF for soft real-time scheduling, they were established assuming a *worst-case* system provisioning: when specifying a task's utilization, which defines its needed processor share, a worst-case execution time is assumed. This is a serious impediment. Indeed, on a multicore platform, worst-case execution costs could conceivably be orders of magnitude greater than average-case costs, making an average- or near-average-case provisioning all but inevitable in many settings. When using Linux, which lacks the determinism of a real-time operating system, a worst-case provisioning is even more questionable. Additionally, timing-analysis tools have not matured enough to effectively determine reasonably tight upper bounds on task execution times on multiprocessors; on the other hand, mean execution times can be easily estimated in an unbiased way from observed data.

In light of these observations, we argue that, if bounded tardiness is acceptable, then the use of a less conservative task execution model should be as well. The natural model for execution times that vary from job to job is a probability distribution. Motivated by this, we present in this paper a derivation of expected tardiness under GEDF when execution times are stochastic. This derivation generalizes that used in the earlier worst-case bound established by Devi and Anderson [1] and places GEDF on a more solid footing as a candidate soft real-time scheduling algorithm.

Related Work in Queueing. It is natural to think about possible parallels between soft real-time scheduling and queueing. Problems with 'due-date' or 'lead-time' requirements have been examined in the queueing literature from a control standpoint, both in admissions and in service disciplines (see, e.g., [2, 8]). In existing work, a single server is assumed, or an asymptotic regime is used where multiple servers reduce to a single server. In this asymptotic regime, an optimal approach for controlling service is the parameterized *generalized longest queue* (GLQ) rule [8]. In a GLQ(θ) discipline, if $N_i(t)$ is the number of customers in the queue of the ith customer type at time t, then service is provided to the customer type with the largest value of $\theta_i N_i(t)$. The GLQ discipline approximates GEDF under the following conditions:

- θ_i is the mean interarrival time of customer type i;

*Work supported by AT&T, IBM, and Sun Corps.; NSF grants CNS 0834270 and CNS 0834132; ARO grant W911NF-09-1-0535; and AFOSR grant FA9550-09-1-0549.

- customer types are indexed in ascending manner by relative deadline; and
- ties are broken by serving the customer type with the smallest index.

The major difference between a queueing system with several customer types and a sporadic task system is that sporadic tasks are sequential: jobs of a single task must execute in order, and they may not overlap. Such precedence constraints do not generally arise in queueing problems.

Nonetheless, the asymptotic optimality of GLQ suggests that GEDF should be a useful algorithm for scheduling sporadic task systems where execution times are stochastic. However, to our knowledge, this issue has not be considered in prior work.

Main Contribution. We derive an expected (mean) tardiness bound under GEDF when task execution costs are defined probabilistically, instead of deterministically; as a byproduct of this analysis, we also obtain bounds on the quantiles (or percentiles) of the tardiness distribution. A major contribution is that our bounds are applicable if the *expected* total utilization is less than the system's capacity. In prior work on worst-case tardiness bounds, a similar requirement is needed from a worst-case perspective. However, we allow the system to be *over*-utilized in the worst case. We assume that execution costs follow a probability distribution, and derive bounds on properties (mean and quantiles) of the tardiness distribution. These bounds depend on the execution-time distributions only through mean, variance, and worst-case execution time.

Organization. The rest of this paper is organized as follows. In Sec. 2, we formalize the system to be studied. In Sec. 3, we show that all jobs complete under GEDF within a bounded number of time units of the time where they complete in a processor-sharing (PS) schedule. In Sec. 4, we show that expected tardiness under a PS schedule is bounded. In Sec. 5, we combine these results to obtain a bound on expected tardiness under GEDF. Finally, we give an example and some remarks about how our result might be extended.

2 System Specification and Properties

In this section, we formalize the task system to be studied. Throughout this paper, we will denote random variables by capital letters, and deterministic quantities by lowercase letters. All time values are continuous.

Task System. A *task system* τ is a collection of sporadic tasks $\{\tau_i, i = 1 \ldots, n\}$. Each *task* τ_i is a possibly infinite sequence of jobs $\{\tau_{i,j}, j = 1, 2, \ldots\}$. A *job* is a segment of code that requires execution on a processor. Jobs must execute sequentially; that is, the next job of a task cannot begin execution until the previous job of that task has completed. We assume that m processors are available to schedule τ.

A task τ_i is specified by its *period* p_i, and its *execution time distribution function* $G_i(x)$, which gives the probability that a job of τ_i requires no more than x time units to execute. We require such a distribution to have finite mean and variance. Any of the standard probability distributions used for modeling, such as uniform, exponential, Weibull, etc., have this property; however, for the analysis to be correct, we also need an upper bound on the expected value of the maximum execution time seen so far at any point in time—a worst-case execution time e_i will suffice. This sufficient condition is the same as assuming that there $\exists e_i < \infty$ such that $G_i(e_i) = 1$.

This specification of tasks is very general. For example, the deterministic case of sporadic task systems often considered in the real-time systems literature, where every task requires e_i time units to complete, is a special case of our system, where $G_i(x)$ is 1 if $x \geq e_i$, and 0 if $x < e_i$.

Schedule-Independent Properties of a Job. The following characteristics of a job $\tau_{i,j}$ do not depend on how jobs are scheduled. The *release time* $r_{i,j}$ is the earliest time that job $\tau_{i,j}$ may execute. We assume the sporadic model, for which $r_{i,1} \geq 0$ and $r_{i,j} \geq r_{i,j-1} + p_i$ for all i and for all $j > 1$. The *deadline* $d_{i,j}$ is the time by which $\tau_{i,j}$ must complete execution. We assume implicit deadlines, i.e., $d_{i,j} = r_{i,j} + p_i$. The *execution time* $X_{i,j}$ is the actual time that $\tau_{i,j}$ executes on a processor, so $\mathsf{P}(X_{i,j} \leq x) = G_i(x)$.

Job $\tau_{i,j}$ is *active* at time t if $r_{i,j} \leq t$ and it has not finished executing. A task is active at time t if any of its jobs are active at time t. Job $\tau_{i,j}$ is *eligible* at time t if $\tau_{i,j}$ is active and its predecessor $\tau_{i,j-1}$ is not active or does not exist.

Each of the above properties depends only on the task specification. We assume that the execution time of $\tau_{i,j}$ is independently and identically distributed according to the distribution $G_i(\cdot)$. This means that the execution of each job of τ_i follows the same distribution, and it does not depend on the execution time of some other job. Note that distributions are not identical across tasks, only jobs of the same task.

Definition (Concrete instance)**.** A *concrete instance* of a task system is an instance where the actual release times and actual execution times are known for each job.

Properties of the Execution Time Distribution. The *expected execution time* of a job of task τ_i is given by

$$\bar{e}_i = \mathsf{E}(X_{i,1}) = \mathsf{E}(X_{i,2}) = \cdots = \int_0^\infty x dG_i(x).$$

The *expected utilization of a task* τ_i is $\bar{u}_i = \bar{e}_i/p_i$. The *expected total utilization of* τ is therefore $\bar{u}_{\text{sum}} = \sum_i \bar{u}_i$ (for background on properties of expectation, see for example, [7, ch. 7]). The *execution time variance* σ_i^2 of a task τ_i is given by $\mathsf{E}\left(X_{i,1}^2\right) - \bar{e}_i^2$.

Schedule-Dependent Properties of a Job. The following characteristics of a job $\tau_{i,j}$ depend on how jobs are scheduled under scheduler S. The *completion time* $C_{i,j}^{\mathsf{S}}$ is the actual time the job completes executing. The *tardiness* $T_{i,j}^{\mathsf{S}}$ is the amount

of time that the job is late:

$$T_{i,j}^{\mathsf{S}} = \max\{C_{i,j}^{\mathsf{S}} - d_{i,j}, 0\}. \quad (1)$$

We will omit the superscript S when it is clear which scheduler is assumed.

Definition (Stability)**.** τ is *stable* if $\bar{u}_{\text{sum}} < m$, where m is the number of processors, and $\bar{u}_i < 1$ for all $\tau_i \in \tau$.

All task systems considered in this paper will be assumed to be stable.

Definition (Schedulability)**.** τ is *schedulable* by a scheduling algorithm S if it can be scheduled by S in such a way that the expected tardiness of every job is bounded.

The following processor-sharing scheduler will be a tool in our analysis.

Definition. PS is a processor-sharing (PS) schedule on m processors where for all $\tau_i \in \tau$, at every instant that τ_i is active, we allocate to τ_i a fraction $\hat{u}_i$ of the processing capacity of one processor, where

$$\sum_{i=1}^{n} \hat{u}_i \leq m \quad (2)$$

$$\bar{u}_i < \hat{u}_i \leq 1, \quad \forall i. \quad (3)$$

At instants when τ_i is not active, it receives no allocation.

When τ_i is active, the fraction of processing capacity allocated to τ_i is thus strictly greater than the fraction needed for it to complete on time in the average case—note that under this model, some jobs may *not* complete by their deadlines in PS. This is a major difference in comparison to how PS schedules are usually defined.

We can imagine that PS is a system of n processors, each of which has a fraction $\hat{u}_i$ of the processing capacity of one processor in the real system, and each of which is dedicated to executing jobs of a specific task. It is important to note that there is not a unique choice of values of $\{\hat{u}_i, i = 1, 2, \ldots, n\}$, and the choice of those values will affect the expected tardiness bound we derive; however, since the stability assumption guarantees that we will always have excess processing capacity, we are guaranteed to be able to find values of $\{\hat{u}_i, i = 1, 2, \ldots, n\}$ satisfying (2) and (3).

Example. Consider the example of a task system with three tasks and two processors, with the following specifications: $(p_1, \bar{e}_1) = (1, 0.8)$, $(p_2, \bar{e}_2) = (2.1, 0.7)$, $(p_3, \bar{e}_3) = (1.6, 0.4)$. Then a feasible choice for $\{\hat{u}_i, i = 1, 2, 3\}$ is $\{0.95, 0.4, 0.65\}$, and the corresponding PS schedule for one instance of this task system is given in Fig. 1 (actual execution times were randomly generated).

Definition. *Global earliest-deadline first* (GEDF) is a schedule on m processors such that: at each time instant where there are more than m active tasks, the m active tasks whose eligible jobs have the earliest deadlines are each allocated one processor; and at each time instant where there are $k \leq m$ active tasks, each active task is allocated one processor, and $m - k$ processors are idled. In the case of deadline ties, a consistent tie-breaker is used. We assume that ties are broken in favor of the task with the smallest index.

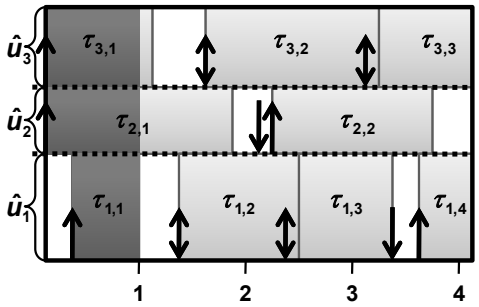

Figure 1: Example PS schedule. The area that is heavily shaded is $a^{\mathsf{PS}}(1)$. Release times are indicated by $\uparrow$ and deadlines are indicated by $\downarrow$.

If the set of release times is fixed, GEDF induces the following ordering on jobs.

Definition (Job Ordering)**.** Given a fixed set of release times, $\tau_{i,j} \prec \tau_{k,l}$ if and only if $d_{i,j} < d_{k,l}$, or $d_{i,j} = d_{k,l}$ and $i < k$.

Definition. The *instantaneous schedule* IS is a schedule where at the time a job is released, it instantaneously receives its full allocation.

PS-induced and GEDF-induced Tardiness. The major result of this paper is that stability implies schedulability under GEDF. In the deterministic model of [1], because there is no uncertainty, a stable system can be scheduled using processor sharing with no tardiness. Therefore, tardiness under GEDF resulted only from the use of the non-optimal GEDF as compared to PS.

Our result generalizes [1] for the case where there may be tardiness under PS. In our model, such tardiness under PS arises due to the uncertainty of execution times. We use IS as a second ideal scheduler because there is no tardiness under it. We will show that the total tardiness of a job under GEDF has two contributions: PS-*induced tardiness*, which comes from the variability of execution costs, and *scheduler-induced tardiness*, due to using a specific non-ideal scheduler (we conduct the analysis for GEDF-induced tardiness).

3 Bounding GEDF-induced Tardiness

In this section, we consider an arbitrary concrete instance of a stable task system, and we show deterministically a relationship between job completion times is GEDF and completion times in PS.

Allocation, Excess Work, and Lag. Define $a_i^{\mathsf{S}}(t)$ to be the *total allocation* to all jobs of τ_i during $[0,t)$ in schedule S. Then $a_i^{\mathsf{GEDF}}(t)$ is the amount of time that jobs of τ_i execute in GEDF in $[0,t)$, because all jobs execute at rate 1.0 in GEDF; $a_i^{\mathsf{PS}}(t)$ is $\hat{u}_i$ times the amount of time that jobs of τ_i execute in GEDF in $[0,t)$, since jobs execute at rate $\hat{u}_i$ in PS; $a_i^{\mathsf{IS}}(t)$ is the sum of the execution times of all jobs of τ_i released prior to t.

For example, in Fig. 1, the heavily shaded area is $a^{\mathsf{PS}}(1)$. Since IS instantly executes each job upon its release, $a^{\mathsf{IS}}(1)$ is the sum of the execution times of $\tau_{1,1}, \tau_{2,1}$, and $\tau_{3,1}$, since they have all been released before time 1. It is always true that $a^{\mathsf{IS}}(t) \geq a^{\mathsf{PS}}(t)$.

We relate the allocation to a task τ_i under PS to its allocation under GEDF using the notion of *lag*, which is defined as

$$\mathcal{L}_i(t) = a_i^{\mathsf{PS}}(t) - a_i^{\mathsf{GEDF}}(t). \tag{4}$$

In addition, we define the *total lag* to be $\mathcal{L}(t) = \sum_i \mathcal{L}_i(t)$.

Similarly, we define the *instantaneous lag* as

$$\mathcal{IL}_i(t) = a_i^{\mathsf{IS}}(t) - a_i^{\mathsf{GEDF}}(t), \tag{5}$$

and *total instantaneous lag* as $\mathcal{IL}(t) = \sum_i \mathcal{IL}_i(t)$. $\mathcal{IL}$ is the same as the amount of remaining work in GEDF, but for our analysis it is more convenient to think about it as being analogous to lag. It is always true that $\mathcal{IL}(t) \geq 0$, since IS gives all needed allocation to a job at the same instant that GEDF can merely *start* working on it.

Definition (Busy and Non-Busy Intervals in GEDF)**.** An interval $\mathcal{I}$ is *busy for* τ_i if some job of τ_i executes in GEDF at each instant of $\mathcal{I}$. An interval $\mathcal{I}$ is a *busy interval* in GEDF if all m processors are utilized at each instant of $\mathcal{I}$. An interval $\mathcal{I}$ is a *non-busy interval* in GEDF if there is at least one idle processor at each instant in $\mathcal{I}$. Note that an interval may be neither busy nor non-busy.

Claim 1. *If interval $\mathcal{I}$ is busy for τ_i, then $\mathcal{L}_i$ is non-increasing over $\mathcal{I}$.*

Proof. If $[t_1, t_2)$ is a busy interval for τ_i, then

$$\begin{aligned}
\mathcal{L}_i(t_2) &= a_i^{\mathsf{PS}}(t_2) - a_i^{\mathsf{GEDF}}(t_2) \\
&\leq a_i^{\mathsf{PS}}(t_1) + \hat{u}_i(t_2 - t_1) - (a_i^{\mathsf{GEDF}}(t_1) + (t_2 - t_1)) \\
&= \mathcal{L}_i(t_1) + (\hat{u}_i - 1)(t_2 - t_1) \\
&\leq \mathcal{L}_i(t_1), \text{ by (3).}
\end{aligned}$$

□

Claim 2. *$\mathcal{L}$ is non-increasing over a busy interval.*

Proof. If $[t_1, t_2)$ is a busy interval, then

$$\begin{aligned}
\mathcal{L}(t_2) &= a^{\mathsf{PS}}(t_2) - a^{\mathsf{GEDF}}(t_2) \\
&\leq a^{\mathsf{PS}}(t_1) + \sum_i \hat{u}_i(t_2 - t_1) - (a^{\mathsf{GEDF}}(t_1) + m(t_2 - t_1)) \\
&\leq \mathcal{L}(t_1) + (\sum_i \hat{u}_i - m)(t_2 - t_1) \\
&\leq \mathcal{L}(t_1), \text{ by (2).}
\end{aligned}$$

□

We derive the result of this section through three steps. First, we show that if some job's completion time does not meet a particular bound (specified later) in GEDF, then there is a lower bound on total instantaneous lag at a certain point in time. This is given in Corollary 4, which appears later. We then show in Lemma 5 that total lag is bounded from above at certain points in time. Finally, in Claim 6, we establish a relationship between total lag and total instantaneous lag. By combining these three results, we upper-bound each job's completion time.

These proofs follow the same structure as the tardiness-bound proof of Devi and Anderson [1]; however, because we allow execution times to vary, deadlines and PS-completion times are no longer identical. This introduces some additional complexity in the proofs (for instance, the schedule IS and instantaneous lag are not used in [1]).

Let τ be a concrete instance of a stable task system and suppose that τ has been scheduled under PS, with fixed $\{\hat{u}_i, i = 1, \ldots, n\}$, and the following information is known for all $\tau_{i,j} \in \tau$:

- release times $r_{i,j}$,
- actual execution costs $e_{i,j}$, and
- PS-completion times $f_{i,j}$.

We use lowercase letters because this is *a posteriori* analysis—all quantities are known. Define

$$\hat{f}_{i,j} = \max\{f_{i,j}, d_{i,j}\} \tag{6}$$

and

$$g_{i,j} = \min\{t \geq d_{i,j} : t \geq f_{k,l}, \forall \tau_{k,l} \preceq \tau_{i,j}\}. \tag{7}$$

Note that

$$g_{i,j} \geq \hat{f}_{i,j} \geq f_{i,j}. \tag{8}$$

Thus, $\hat{f}_{i,j}$ is the earliest point in time at or after $\tau_{i,j}$'s deadline by which $\tau_{i,j}$ has completed in PS, and $g_{i,j}$ is the earliest point in time at or after $\tau_{i,j}$'s deadline where all work of priority higher than or equal to $\tau_{i,j}$ has completed in PS.

We observe that job $\tau_{i,j}$'s tardiness in PS equals $\hat{f}_{i,j} - d_{i,j}$ and is no more than $g_{i,j} - d_{i,j}$.

Define $\hat{e}_{i,j} = \max_{j' \leq j}\{e_{i,j'}\}$.

We first show that if the completion time of job $\tau_{i,j}$ in GEDF is later than a certain bound, then there is a lower bound on instantaneous lag at $\hat{f}_{i,j}$.

Let $c_{k,l}$ denote the completion time of job $\tau_{k,l}$ in GEDF.

Lemma 3. *Let $\xi \geq 0$. Suppose that for all $\tau_{k,l} \prec \tau_{i,j}$,*

$$c_{k,l} \leq \hat{f}_{k,l} + \xi + \hat{e}_{k,l}. \tag{9}$$

If

$$\mathcal{IL}(\hat{f}_{i,j}) \leq m\xi + \hat{e}_{i,j}, \tag{10}$$

then

$$c_{i,j} \leq \hat{f}_{i,j} + \xi + \hat{e}_{i,j}. \tag{11}$$

Proof. For this proof, we ignore all jobs $\tau_{p,q} \succ \tau_{i,j}$ (i.e., assume they are not present in any schedule considered in the proof) because they cannot preempt $\tau_{i,j}$ in GEDF and thus they cannot delay its completion time. This assumption does not cause any loss of generality—it simply reduces clutter in the proof.

Consider the point in time $b_{i,j} = \max\{c_{i,j-1}, v_{i,j}\}$, where

$$v_{i,j} = \min\{t \geq \hat{f}_{i,j} : [t, \infty) \text{ is a non-busy interval}\}.$$

At $b_{i,j}$, $\tau_{i,j}$ must have begun executing in GEDF, because it is eligible (since $b_{i,j} \geq v_{i,j} \geq \hat{f}_{i,j} \geq d_{i,j} \geq r_{i,j}$), its predecessor has completed (since $b_{i,j} \geq c_{i,j-1}$), and there is an idle processor (GEDF is work-conserving). Furthermore, since $b_{i,j} \geq d_{i,j}$, no new jobs will be released that could preempt $\tau_{i,j}$. Therefore, $\tau_{i,j}$ completes by time $b_{i,j} + e_{i,j}$, or in other words,

$$\begin{aligned} c_{i,j} &\leq \max\{c_{i,j-1} + e_{i,j}, v_{i,j} + e_{i,j}\} \\ &\leq \max\{\hat{f}_{i,j-1} + \xi + \hat{e}_{i,j-1} + e_{i,j}, v_{i,j} + e_{i,j}\}, \text{ by (9).} \end{aligned} \tag{12}$$

Now, because jobs of τ_i execute sequentially at a rate of $\hat{u}_i$ in PS, and do not begin until their predecessors complete, we have

$$\begin{aligned} f_{i,j} &\geq f_{i,j-1} + e_{i,j}/\hat{u}_i \\ &\geq f_{i,j-1} + e_{i,j}, \text{ by (3),} \end{aligned} \tag{13}$$

and because jobs do not begin executing in PS before their release times, we have

$$\begin{aligned} f_{i,j} &\geq r_{i,j} + e_{i,j}/\hat{u}_i \\ &\geq r_{i,j} + e_{i,j}, \text{ by (3)} \\ &\geq d_{i,j-1} + e_{i,j}, \text{ since tasks are sporadic.} \end{aligned} \tag{14}$$

Combining (13) with (14), we get $f_{i,j} \geq \max\{f_{i,j-1}, d_{i,j-1}\} + e_{i,j} = \hat{f}_{i,j-1} + e_{i,j}$. Then (12) becomes

$$\begin{aligned} c_{i,j} &\leq \max\{f_{i,j} + \xi + \hat{e}_{i,j-1}, v_{i,j} + e_{i,j}\} \\ &\leq \max\{\hat{f}_{i,j} + \xi + \hat{e}_{i,j-1}, v_{i,j} + e_{i,j}\}, \end{aligned} \tag{15}$$

using the definition of $\hat{f}_{i,j}$.

Since $\hat{e}_{i,j-1} \leq \hat{e}_{i,j}$ and $e_{i,j} \leq \hat{e}_{i,j}$, if $v_{i,j} \leq \hat{f}_{i,j} + \xi$, then (11) holds, so assume otherwise, i.e., $v_{i,j} > \hat{f}_{i,j} + \xi$. In this case, $[\hat{f}_{i,j}, \hat{f}_{i,j} + \xi)$ is a busy interval. Since by definition, all work is completed in IS by $\hat{f}_{i,j}$ (since $\hat{f}_{i,j} \geq d_{i,j}$ and no jobs are released after $d_{i,j}$) and $\mathcal{IL}(\hat{f}_{i,j}) \leq m\xi + \hat{e}_{i,j}$ by (10), there must be at most $m\xi + \hat{e}_{i,j}$ work to complete in GEDF after $\hat{f}_{i,j}$. Since $[\hat{f}_{i,j}, \hat{f}_{i,j} + \xi)$ is a busy interval, in which $m\xi$ work is completed in GEDF, at time $\hat{f}_{i,j} + \xi$, there must be at most $\hat{e}_{i,j}$ work to complete in GEDF. Even if all this work executes sequentially, it will be complete by $\hat{f}_{i,j} + \xi + \hat{e}_{i,j}$, and hence (11) holds. □

Corollary 4. *Given $\xi \geq 0$, let $\tau_{i,j}$ be the minimal job under $\prec$ such that $c_{i,j} > \hat{f}_{i,j} + \xi + \hat{e}_{i,j}$. Then $\mathcal{IL}(\hat{f}_{i,j}) > m\xi + \hat{e}_{i,j}$.*

Proof. This is the contrapositive statement of Lemma 3, so it follows immediately. □

For the next lemma, we will need the concept of a maximal non-busy subinterval.

Definition (Maximal Non-Busy Subinterval)**.** An interval $[t_1, t_2)$ is a *maximal non-busy subinterval in* $[0, t)$ if and only if $[t_2, t)$ is a busy interval, $[t_1, t_2)$ is a non-busy interval, and $\forall t_0 < t_1$, $[t_0, t_2)$ is not a non-busy interval.

If $[0, t)$ is busy, then the maximal non-busy subinterval in $[0, t)$ is empty. If $[0, t)$ is non-busy then the maximal non-busy subinterval in $[0, t)$ is $[0, t)$ itself.

We now show that there is an upper bound on total lag at time $g_{i,j}$.

Let $l_k = \max\{l : \tau_{k,l} \preceq \tau_{i,j}\}$. Define

$$\upsilon = \sum_{\tau_k \in \mathcal{U}_{\max}} \hat{u}_k, \tag{16}$$

where $\mathcal{U}_{\max}$ is the set of $m - 1$ tasks with largest values of $\{\hat{u}_i\}$, and define

$$\eta_{i,j} = \sum_{\tau_k \in \mathcal{E}_{\max}} \hat{e}_{k,l_k}, \tag{17}$$

where $\mathcal{E}_{\max}$ is the set of $m - 1$ tasks with largest values of $\hat{e}_{k,l_k}$.

Lemma 5. *For some $\xi \geq 0$, suppose that for all $\tau_{k,l} \prec \tau_{i,j}$, $c_{k,l} \leq g_{k,l} + \xi + \hat{e}_{k,l}$. Then $\mathcal{L}(g_{i,j}) \leq \upsilon\xi + \eta_{i,j}$.*

Proof. We once again ignore any jobs $\tau_{p,q} \succ \tau_{i,j}$, to reduce clutter in the proof (that is, we assume that no such job appears in any schedule under consideration).

Let $[t_1, t_2)$ be the maximal non-busy subinterval in $[0, g_{i,j})$ in GEDF. If such an interval does not exist then $[0, g_{i,j})$ is busy, and by Claim 2, $\mathcal{L}(g_{i,j}) \leq 0$.

For each $\tau_k \in \tau$ for which some job is executing at t_1 in GEDF, denote that job by τ_{k,l_k}.

We define the superscript "$-$" to be shorthand for the left limit: $t^- = \lim_{\epsilon \to 0^+} t - \epsilon$. Further, we define

$$\alpha_{k,l} = \sum_{l' \leq l} e_{k,l'}$$

to be the *total required allocation* for jobs of τ_k up to and including $\tau_{k,l}$. Since all jobs of τ_i up to and including $\tau_{k,l}$ have completed by $f_{k,l}$ in PS,

$$a_k^{\mathsf{PS}}(f_{k,l}) = \alpha_{k,l}, \tag{18}$$

and hence by (8),

$$a_k^{\mathsf{PS}}(g_{k,l}) = \alpha_{k,l}. \tag{19}$$

For any time t and any job $\tau_{k,l}$, define $\delta_{k,l}(t)$ to be the amount of work completed in GEDF on job $\tau_{k,l}$ prior to time t.

If $\tau_{k,l}$ is executing at time t in GEDF, then

$$a_k^{\mathsf{GEDF}} = \alpha_{k,l} - e_{k,l} + \delta_{k,l}(t), \tag{20}$$

since τ_k requires only $e_{k,l} - \delta_{k,l}(t)$ more allocation before it will have had its total required allocation.

Now partition the tasks of τ into four sets:

Set 1. Tasks that do not have a job executing at t_2^- in GEDF.

Set 2. Tasks that have a job executing at t_2^- in GEDF but for which $[t_1, t_2)$ is not busy.

Set 3. Tasks for which $[t_1, t_2)$ is busy and $t_1 \leq g_{k,l_k}$.

Set 4. Tasks for which $[t_1, t_2)$ is busy and $t_1 > g_{k,l_k}$.

Note that Sets 3 and 4 can together include at most $m-1$ tasks, because $[t_1, t_2)$ is non-busy.

We now give an upper bound on $\mathcal{L}_k(t_2)$ for all $\tau_k \in \tau$, and then sum to obtain an upper bound on $\mathcal{L}(t_2)$. Since $[t_2, g_{i,j})$ is a busy interval, the result will also be an upper bound on $\mathcal{L}(g_{i,j})$ by Claim 2.

Set 1. τ_k does not have a job executing at t_2^- in GEDF. Then no work remains to be done on τ_k in GEDF at t_2^-, since there is an idle processor and GEDF is work-conserving. Therefore, $\mathcal{L}_k(t_2) \leq 0$.

Set 2. τ_k does have a job executing at t_2^- in GEDF but $[t_1, t_2)$ is not busy for τ_k. Then $\exists t_0 \in [t_1, t_2)$, such that in GEDF, τ_k does not execute at t_0^- but does execute at t_0. Choose t_0 to be the latest such time instant. Then t_0 must be the release time of some job of τ_k, since there is an idle processor and GEDF is work-conserving. Thus, $\mathcal{L}_k(t_0) \leq 0$ and since $[t_0, t_2)$ is busy for τ_k, $\mathcal{L}_k(t_2) \leq \mathcal{L}_k(t_0) \leq 0$, by Claim 2.

Set 3. $[t_1, t_2)$ is busy for τ_k and $t_1 \leq g_{k,l_k}$. Then

$$\begin{aligned}
\mathcal{L}_k(t_2) &\leq \mathcal{L}_k(t_1), \text{ by Claim 1} \\
&= a_k^{\mathsf{PS}}(t_1) - a_k^{\mathsf{GEDF}}(t_1), \text{ by (4)} \\
&\leq a_k^{\mathsf{PS}}(g_{k,l_k}) - a_k^{\mathsf{GEDF}}(t_1), \text{ since } t_1 \leq g_{k,l_k} \\
&= \alpha_{k,l_k} - a_k^{\mathsf{GEDF}}(t_1), \text{ by (19)} \\
&= \alpha_{k,l_k} - (\alpha_{k,l_k} - e_{k,l_k} + \delta_{k,l_k}(t_1)), \text{ by (20)} \\
&\leq e_{k,l_k} \\
&\leq \hat{e}_{k,l_k}.
\end{aligned} \tag{21}$$

Set 4. $[t_1, t_2)$ is busy for τ_k and $t_1 > g_{k,l_k}$. Note that by the definition of $[t_1, t_2)$, $t_1 < g_{i,j}$ and therefore, τ_i cannot be in Set 4. For any task τ_k in this set, we have

$$\begin{aligned}
\mathcal{L}_k(t_2) &\leq \mathcal{L}_k(t_1), \text{ by Claim 1} \\
&= a_k^{\mathsf{PS}}(t_1) - a_k^{\mathsf{GEDF}}(t_1), \text{ by (4)} \\
&\leq a_k^{\mathsf{PS}}(g_{k,l_k}) + \hat{u}_k(t_1 - g_{k,l_k}) - a_k^{\mathsf{GEDF}}(t_1), \\
&\quad \text{since } \tau_k \text{ executes at rate} \leq \hat{u}_k \text{ in } [t_1, g_{k,l_k}) \\
&= \alpha_{k,l_k} + \hat{u}_k(t_1 - g_{k,l_k}) \\
&\quad - (\alpha_{k,l_k} - e_{k,l_k} + \delta_{k,l_k}(t_1)), \text{ by (19),(20)} \\
&\leq \hat{u}_k(t_1 - g_{k,l_k}) + e_{k,l_k} - \delta_{k,l_k}(t_1).
\end{aligned} \tag{22}$$

Since τ_{k,l_k} is executing in GEDF at t_1, $t_1 + e_{k,l_k} - \delta_{k,l_k}(t_1) \leq c_{k,l_k}$, and therefore

$$\begin{aligned}
t_1 - g_{k,l_k} &\leq c_{k,l_k} - g_{k,l_k} + \delta_{k,l_k}(t_1) - e_{k,l_k} \\
&\leq c_{k,l_k} - \hat{f}_{k,l_k} + \delta_{k,l_k}(t_1) - e_{k,l_k}, \text{ by (8)} \\
&\leq \xi + \hat{e}_{k,l_k} + \delta_{k,l_k}(t_1) - e_{k,l_k},
\end{aligned}$$

where the last inequality follows by the assumption of the lemma (as noted above, τ_i is not in Set 4, so we can apply that assumption here). Combining this result with (22) yields

$$\begin{aligned}
\mathcal{L}_k(t_2) &\leq \hat{u}_k(\xi + \hat{e}_{k,l_k}) + (1 - \hat{u}_k)(e_{k,l_k} - \delta_{k,l_k}(t_1)) \\
&\leq \hat{u}_k(\xi + \hat{e}_{k,l_k}) + (1 - \hat{u}_k)(\hat{e}_{k,l_k} - \delta_{k,l_k}(t_1)) \\
&= \hat{u}_k\xi + \hat{e}_{k,l_k} + (\hat{u}_k - 1)\delta_{k,l_k}(t_1) \\
&\leq \hat{u}_k\xi + \hat{e}_{k,l_k}, \text{ by (3).}
\end{aligned} \tag{23}$$

Summing Over All Tasks. We showed that tasks in Sets 1 and 2 have non-positive lag, so to give an upper bound on $\mathcal{L}(t_2)$, we need to only sum $\hat{u}_k\xi + \hat{e}_{k,l_k}$ over the $m-1$ most expensive values of $\hat{u}_k$ and $\hat{e}_{k,l_k}$ respectively. Then as noted earlier, $\mathcal{L}(g_{i,j}) \leq \mathcal{L}(t_2)$, and

$$\mathcal{L}(t_2) \leq \upsilon\xi + \eta_{i,j}. \qquad \square$$

We now establish a relationship between lag and instantaneous lag.

Claim 6. *Let $\tau_{i,j}$ be the maximal job under $\prec$. Then $\mathcal{IL}(\hat{f}_{i,j}) \leq \mathcal{L}(g_{i,j}) + m(g_{i,j} - \hat{f}_{i,j})$.*

Proof. At $g_{i,j}$ all work on jobs of priority higher than or equal to $\tau_{i,j}$ has completed in PS. At $\hat{f}_{i,j}$ all work on jobs of priority higher than or equal to $\tau_{i,j}$ has completed in IS. Therefore,

$$a^{\mathsf{PS}}(g_{i,j}) = a^{\mathsf{IS}}(\hat{f}_{i,j}). \tag{24}$$

Since jobs execute at a total rate of no more than m in GEDF, we also have

$$a^{\mathsf{GEDF}}(g_{i,j}) \leq a^{\mathsf{GEDF}}(\hat{f}_{i,j}) + m(g_{i,j} - \hat{f}_{i,j}). \tag{25}$$

Thus, we have

$$\begin{aligned}
\mathcal{IL}(\hat{f}_{i,j}) &= a^{\mathsf{IS}}(\hat{f}_{i,j}) - a^{\mathsf{GEDF}}(\hat{f}_{i,j}) \\
&= a^{\mathsf{PS}}(g_{i,j}) - a^{\mathsf{GEDF}}(\hat{f}_{i,j}), \text{ by (24)} \\
&\leq a^{\mathsf{PS}}(g_{i,j}) - a^{\mathsf{GEDF}}(g_{i,j}) + m(g_{i,j} - \hat{f}_{i,j}), \text{ by (25)} \\
&= \mathcal{L}(g_{i,j}) + m(g_{i,j} - \hat{f}_{i,j}), \text{ by (4).}
\end{aligned} \qquad \square$$

Theorem 7. *Let* $\xi^*_{i,j} = \frac{\eta_{i,j}+m(g_{i,j}-\hat{f}_{i,j})}{m-\upsilon}$. *For all jobs* $\tau_{i,j}$ *in any concrete instance of* τ, $c_{i,j} \leq \hat{f}_{i,j} + \xi^*_{i,j} + \hat{e}_{i,j}$.

Proof. Suppose not. Then there is a concrete instance of τ where $\tau_{i,j}$ is the minimal job under $\prec$ such that $c_{i,j} > \hat{f}_{i,j} + \xi^*_{i,j} + \hat{e}_{i,j}$, and $c_{k,l} \leq \hat{f}_{k,l} + \xi^*_{i,j} + \hat{e}_{k,l}$ for all $\tau_{k,l} \prec \tau_{i,j}$. Then fix as a constant $\xi = \xi^*_{i,j}$. By Corollary 4, $\mathcal{IL}(\hat{f}_{i,j}) > m\xi + \hat{e}_{i,j}$. By Lemma 5, $\mathcal{L}(g_{i,j}) \leq \upsilon\xi + \eta_{i,j}$. By combining these two statements and the result of Claim 6, $m\xi+\hat{e}_{i,j} < \upsilon\xi+\eta_{i,j}+m(g_{i,j}-\hat{f}_{i,j})$, and solving for ξ yields $\xi < \frac{\eta_{i,j}+m(g_{i,j}-\hat{f}_{i,j})-\hat{e}_{i,j}}{m-\upsilon}$, which is a contradiction. □

So we have shown that all jobs complete in GEDF within a certain number of time units of their deadline or their completion time in PS, whichever is later.

4 Bounding PS-induced Tardiness

In this section, we derive a bound on the expected (mean) value of PS-induced tardiness; in particular, we show that the expected tardiness of any job under the PS schedule can be bounded from above by a constant that depends only on the task's period, the execution-time distributions of the tasks, and the values of $\{\hat{u}_i\}$.

Theorem 8. *There exists a constant* ψ *such that for all* (i,j) *where* $\tau_{i,j}$ *is a job of* τ,

$$\mathsf{E}\left(T^{\mathsf{PS}}_{i,j}\right) \leq \hat{u}_i\psi. \tag{26}$$

The value of ψ *depends on the execution-time distributions of the tasks in* τ, *and on the chosen values of* $\{\hat{u}_i\}$.

To prove this result, we model the tardiness of a job in PS as a stochastic process, and then we use a result from queueing theory to give the bound.

Stochastic Model and Limiting Result. We consider each task individually, since jobs of different tasks do not affect one another in PS.

Choose some task τ_i and examine its jobs $\tau_{i,j}$, for $j = 1, 2, \ldots$. We can write a recursion for $T_{i,j}$ (omitting the superscript PS) as follows: tardiness at the deadline of job $\tau_{i,j}$ is no more than the existing tardiness at the deadline of job $\tau_{i,j-1}$, *plus* the amount of time job $\tau_{i,j}$ needs to execute in PS, which is $X_{i,j}/\hat{u}_i$ (since it executes at rate $\hat{u}_i$), *minus* the amount of time between its release and its deadline (i.e., its period):

$$\begin{aligned} &T_{i,j} \leq \max\{0, T_{i,j-1} + X_{i,j}/\hat{u}_i - p_i\} \forall j \geq 1 \\ &T_{i,0} = 0. \end{aligned} \tag{27}$$

The max is present simply to ensure that $T_{i,j}$ is never negative.

This recursion can be visualized in Fig. 2: By examining the time between the release and deadline of the second job

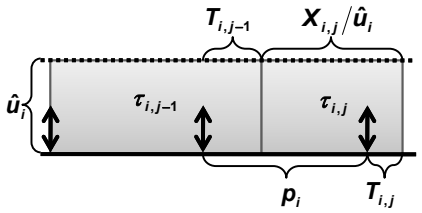

Figure 2: Illustration of the recursion for tardiness in PS.

in the figure, we see that $p_i = T_{i,j-1} + X_{i,j}/\hat{u}_i - T_{i,,}$, or in other words, $T_{i,j} = T_{i,j-1} + X_{i,j}/\hat{u}_i - p_i$.

The recursion (27) is identical to the recursive equation of the waiting time process in a single server G/G/1 queue. Such processes, and the expression (27), were first studied in 1952 by Lindley [5].

Lemma 9. *The stochastic process* $\{T_{i,j}, j \geq 1\}$ *has a limit; in other words, it approaches a single random variable* T_i, *which has a probability distribution function* $\pi_i(\cdot)$ *as* j *increases.*

Proof. A theorem of [5] establishes that as j increases, the distribution of $T_{i,j}$ has a limit if and only if

$$\mathsf{E}\left(Y_{i,j} - Z_{i,j}\right) < 0,$$

where $Y_{i,j}$ is the service time and $Z_{i,j}$ is the interarrival time between customers. To match this result about queues with (27), we set $Y_{i,j} = X_{i,j}/\hat{u}_i$ and let $Z_{i,j}$ be a random variable representing the inter-release time of jobs; since the sporadic task model is assumed, $Z_{i,j} \geq p_i$, $\forall j$, and therefore,

$$\begin{aligned} \mathsf{E}\left(Y_{i,j} - Z_{i,j}\right) &= \mathsf{E}\left(X_{i,j}/\hat{u}_i - Z_{i,j}\right) \\ &\leq \bar{e}_i/\hat{u}_i - p_i, \end{aligned}$$

which is negative by (3). □

Furthermore, from [3, p. 474], we have the following result:

$$\mathsf{E}\left(T_i\right) \leq \frac{\sigma_i^2}{2(p_i - \bar{e}_i/\hat{u}_i)}, \tag{28}$$

where σ_i^2 is the variance of the execution time of τ_i.

The above results therefore guarantee that as j gets large, $T_{i,j}$ approaches T_i, a random variable with a finite mean.

An Equilibrium Result. It is not enough to have $T_{i,j}$ bounded in expectation for only large j; we instead wish to have it bounded for all j.

It should be clear that $T_{i,j}$ is a Markov process (its future evolution depends on its history only through its present state): if we examine (27), we see that $T_{i,j+1}$ depends only on $T_{i,j}$, some constants, and $X_{i,j}$, which is independent of everything else. Therefore, $T_{i,j+1}$ does not depend on $T_{i,j-1}, T_{i,j-2}$, etc.

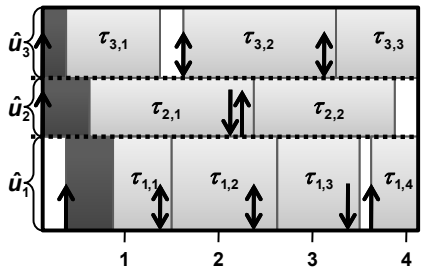

Figure 3: $\tilde{\mathsf{PS}}$ schedule for the example of Figure 1. The shaded areas represent delays drawn from the limiting distributions.

The kernel of a Markov process [6, p. 59] is a probability transition function $K(\cdot)$ that satisfies

$$\pi_{i,j+1}(x) = \int K_i(x,y)\pi_{i,j}(y)dy. \tag{29}$$

Here, $\pi_{i,j}(x)$ is the probability density function of $T_{i,j}$. As j increases, $T_{i,j}$ approaches its limit T_i (by Lemma 9). T_i has probability density function $\pi_i(x)$, which does not depend on j. Therefore, $\pi_i(x)$ satisfies

$$\pi_i(x) = \int K_i(x,y)\pi_i(y)dy \tag{30}$$

So, if we could draw $T_{i,0}$ from the limiting distribution $\pi_i(x)$ instead of fixing it to be zero in (27), then (29)-(30) would guarantee that

$$\pi_{i,0}(x) = \pi_{i,1}(x) = \pi_{i,2}(x) = \cdots = \pi_i(x). \tag{31}$$

Then, the expected PS tardiness bound would apply for all j, not just in the limit. We show that it is possible to do this without interfering with any of the results of this paper.

Definition ($\tilde{\mathsf{PS}}$)**.** The schedule $\tilde{\mathsf{PS}}$ is identical to PS except for the following modification. For each task τ_i, draw a value $T_{i,0}$ from its limiting distribution $\pi_i(\cdot)$. Begin execution of $\tau_{i,1}$ in $\tilde{\mathsf{PS}}$ at a time $T_{i,0}$ units after $\tau_{i,1}$ is released.

Since the execution of the first job of each task is delayed by a non-negative amount of time, the first job of each task begins executing in $\tilde{\mathsf{PS}}$ at or after the time it begins executing in PS, and thus completes in $\tilde{\mathsf{PS}}$ at or after the time it completes in PS. Furthermore, if $C^{\tilde{\mathsf{PS}}}_{i,j-1} \geq C^{\mathsf{PS}}_{i,j-1}$, then $C^{\tilde{\mathsf{PS}}}_{i,j} \geq C^{\mathsf{PS}}_{i,j}$, because in either schedule $\tau_{i,j}$ begins executing at $r_{i,j}$ or at $C^{\mathsf{S}}_{i,j-1}$, whichever is later (e.g., Figs. 1 and 3). By this induction argument, no job completes in $\tilde{\mathsf{PS}}$ before it completes in PS , and thus for all $\tau_{i,j} \in \tau$, $C^{\tilde{\mathsf{PS}}}_{i,j} \geq C^{\mathsf{PS}}_{i,j}$, and therefore

$$T^{\mathsf{PS}}_{i,j}(t) \leq T^{\tilde{\mathsf{PS}}}_{i,j}(t). \tag{32}$$

Since $T^{\tilde{\mathsf{PS}}}_{i,j}$ has distribution $\pi_i(\cdot)$ for all j, it has $\mathsf{E}\left(T^{\tilde{\mathsf{PS}}}_{i,j}\right) = \mathsf{E}\left(T_i\right) \leq \frac{\sigma_i^2}{2(p_i - \bar{e}_i/\hat{u}_i)}$, by (28), and so $\forall j$,

$$\mathsf{E}\left(T^{\mathsf{PS}}_{i,j}\right) \leq \frac{\sigma_i^2}{2(p_i - \bar{e}_i/\hat{u}_i)}, \text{ by (32).} \tag{33}$$

This bound for expected tardiness for any given task will apply to all of its jobs.

Finding Values for $\{\hat{u}_i\}$ and ψ. We still need a value of ψ that satisfies the bound (26) given in Theorem 8 (thereby proving the theorem); specifically, we need $\psi \geq \mathsf{E}\left(T^{\mathsf{PS}}_{i,j}\right)\frac{1}{\hat{u}_i}$, $\forall i$. By (33), it is sufficient to take any ψ such that

$$\psi \geq \max_i \left\{ \frac{\sigma_i^2}{2(\hat{u}_i p_i - \bar{e}_i)} \right\}, \tag{34}$$

but it is not possible to simply take the smallest value of ψ satisfying the above inequalities, because the values of $\{\hat{u}_i\}$ are not fixed—they can be set to any values that satisfy (2)-(3), which are required for the system to be stable. We need to simultaneously determine values for $\{\hat{u}_i\}$ and ψ. To do this, we re-write the problem of choosing the smallest ψ as the problem of choosing the largest ψ^{-1} such that

$$\psi^{-1} \leq \min_i \left\{ \frac{2(\hat{u}_i p_i - \bar{e}_i)}{\sigma_i^2} \right\}. \tag{35}$$

It is possible to express these requirements in a linear fashion: set $\zeta \equiv \psi^{-1}$ and $\{\hat{u}_i, i = 1, \ldots, n\}$ as decision variables; then, an optimal solution to the following linear program will give us a valid combination of allocations and an upper bound for the expected tardiness in PS.

$$\begin{aligned}
\max \quad & \zeta && \text{(LP)}\\
\text{s.t.} \quad & p_i\hat{u}_i - \frac{\sigma_i^2}{2}\zeta \geq \bar{e}_i \quad \forall i && \text{(C1)}\\
& \sum_{i=1}^{n} \hat{u}_i \leq m && \text{(C2)}\\
& \bar{u}_i \leq \hat{u}_i \leq 1 \quad \forall i && \text{(C3)}
\end{aligned}$$

Constraint set (C1) enforces (35), constraint (C2) enforces (2), and constraint set (C3) enforces (3), with one exception—in linear programming we cannot use strict inequalities, so we may have a feasible solution where $\hat{u}_i = \bar{u}_i$ for some i, which is not allowed because it violates (3), which is used in the proof of Lemma 9. However, by examining the first set of constraints, it is easy to see that $\hat{u}_i = \bar{u}_i$ will only occur in the *optimal* solution for tasks where $\sigma_i^2 = 0$, since we are maximizing ζ. $\sigma_i^2 = 0$ is the deterministic case, where we do not need to rely on Lemma 9, because the tardiness of τ_i in PS is always zero in that case.

Proof of Theorem 8. Fix $\{u_i^*\}, \zeta^*$ to be the optimal solution to (LP), and $\psi = \frac{1}{\zeta^*}$. Then, as we established above, ψ satisfies (26), and the theorem is proved. □

5 Final Result

In Sec. 3, we showed that each job always completes in GEDF within a certain amount of time from $\hat{f}_{i,j}$. In Sec. 4, we showed that there is a bound on expected tardiness in PS.

We now combine these two results to show that tardiness in GEDF is bounded in expectation.

This analysis is *a priori*, so we must now treat some quantities as random variables, which will be denoted by capital letters: let $\hat{X}_{i,j}$ be the stochastic equivalent of $\hat{e}_{i,j}$, i.e., $\hat{X}_{i,j} = \max_{j'<j} X_{i,j'}$; let $E_{i,j}$ be the stochastic equivalent of $\eta_{i,j}$, i.e., $E_{i,j}$ is the sum of the $m-1$ largest values of $\hat{X}_{k,l_k}$.

Corollary 10. *Let τ be a stable sporadic task system. Then the expected (mean) tardiness of every job $\tau_{i,j}$ in* GEDF *is no more than*

$$\beta_{i,j} = \hat{u}_i\psi + \frac{\mathsf{E}\left(E_{i,j}\right) + m^2\psi}{m-\upsilon} + \mathsf{E}\left(\hat{X}_{i,j}\right), \tag{36}$$

where $\psi = 1/\zeta^*$ with ζ^* being the optimal solution to (LP). Furthermore, the q-quantile of the tardiness of every job $\tau_{i,j}$ is no more than

$$\frac{1}{1-q}\beta_{i,j}. \tag{37}$$

Proof. From Theorem 7, we have that

$$c_{i,j} \le \hat{f}_{i,j} + \frac{\eta_{i,j} + m(g_{i,j} - \hat{f}_{i,j})}{m-\upsilon} + \hat{e}_{i,j}.$$

Since $\hat{f}_{i,j} \ge d_{i,j}$ by (6),

$$\begin{aligned} T_{i,j}^{\mathsf{GEDF}} &= \max\{c_{i,j} - d_{i,j}, 0\} \\ &\le \hat{f}_{i,j} - d_{i,j} + \frac{\eta_{i,j} + m(g_{i,j} - \hat{f}_{i,j})}{m-\upsilon} + \hat{e}_{i,j}. \end{aligned} \tag{38}$$

The quantity $\hat{f}_{i,j} - d_{i,j}$, the time from job $\tau_{i,j}$'s deadline until it is completed in PS, is not known *a priori*. But since (38) holds no matter what the value of $\hat{f}_{i,j} - d_{i,j}$ is, $\hat{f}_{i,j} - d_{i,j}$ is nothing more than the random variable $T_{i,j}^{\mathsf{PS}}$. Similarly, we will treat $H_{i,j} = g_{i,j} - d_{i,j}$ as a random variable, since the time from $d_{i,j}$ until all higher or equal priority work is completed in PS is not known *a priori*. We do the same with $\hat{e}_{i,j}$ and $\eta_{i,j}$. By using the facts that $g_{i,j} - \hat{f}_{i,j} = g_{i,j} - d_{i,j} - (\hat{f}_{i,j} - d_{i,j})$ and $T_{i,j}^{\mathsf{PS}}$ is non-negative, and taking the expectation of both sides of (38), we obtain

$$\mathsf{E}\left(T_{i,j}^{\mathsf{GEDF}}\right) \le \mathsf{E}\left(T_{i,j}^{\mathsf{PS}}\right) + \frac{\mathsf{E}\left(E_{i,j}\right) + m\mathsf{E}\left(H_{i,j}\right)}{m-\upsilon} + \mathsf{E}\left(\hat{X}_{i,j}\right). \tag{39}$$

For each $\tau_k \in \tau$, define $l_k = \max\{l : \tau_{k,l} \preceq \tau_{i,j}\}$; then, since $d_{k,l_k} \le d_{i,j}$, $\forall \tau_{k,l_k} \preceq \tau_{i,j}$, $H_{i,j} \le \max_k\{T_{k,l_k}^{\mathsf{PS}}\}$, so

$$\begin{aligned} \mathsf{E}\left(H_{i,j}\right) &\le \mathsf{E}\left(\max_k\{T_{k,l_k}^{\mathsf{PS}}\}\right) \\ &\le \mathsf{E}\left(\sum_k T_{k,l_k}^{\mathsf{PS}}\right) \\ &\le \sum_k \mathsf{E}\left(T_{k,l_k}^{\mathsf{PS}}\right) \\ &\le m\psi, \text{ by Theorem 8.} \end{aligned} \tag{40}$$

By Theorem 8, $\mathsf{E}\left(T_{i,j}^{\mathsf{PS}}\right) \le \hat{u}_k\psi$; combined with (40) and (39), this yields (36).

To prove the quantile result, let k be the q-quantile of $T_{i,j}^{\mathsf{GEDF}}$; that is, $\mathsf{P}\left(T_{i,j}^{\mathsf{GEDF}} \le k\right) = q$. Since $T_{i,j}^{\mathsf{GEDF}}$ is a non-negative random variable, Markov's inequality [7, p. 400] states that for any a,

$$\mathsf{P}\left(T_{i,j}^{\mathsf{GEDF}} > a\right) \le \frac{\mathsf{E}\left(T_{i,j}^{\mathsf{GEDF}}\right)}{a}.$$

Letting $a = k$, we have

$$(1-q)k \le \mathsf{E}\left(T_{i,j}^{\mathsf{GEDF}}\right) \le \beta_{i,j},$$

and the result follows immediately. □

Corollary 11. *If worst-case execution times $\{e_i, \tau_i \in \tau\}$ exist, then $\beta_{i,j}$ is upper-bounded by a constant for all i, j.*

Proof. Since each task has a worst-case execution time e_i, then $\mathsf{E}\left(\hat{X}_{i,j}\right) \le e_i$ for all $\tau_{i,j} \in \tau$, and

$$\mathsf{E}\left(E_{i,j}\right) \le \sum_{\tau_k \in \mathcal{E'}_{\max}} e_i$$

for all $\tau_{i,j}$, where $\mathcal{E'}_{\max}$ is the set of $m-1$ tasks with largest values of e_i. □

The quantities that appear in the final bound in (36) only involve ψ, $\{\hat{u}_i\}$, and $\mathsf{E}\left(\hat{X}_{i,j}\right)$. The values of ψ and $\{\hat{u}_i\}$ are obtained from the LP, whose specification involves only means and variances of the execution-time distributions. For the bound on $\mathsf{E}\left(\hat{X}_{i,j}\right)$, worst-case execution times are sufficient, but it might be possible to compute a tighter bound; for example, if the worst-case execution time has an extremely low probability of occurrence, then $\mathsf{E}\left(\hat{X}_{i,j}\right)$ may be much smaller than e_i for small j. We leave as an open problem whether an upper bound on expected tardiness can be expressed entirely in terms of means and variances, without worst-case execution times or information about the distributions.

The quantile result is important because it allows a system designer to determine a bound that will hold any desired percentage of time (e.g., the 0.9-quantile is a threshold that will be met in at least 90% of cases). From a practical standpoint, this allows the more conservative user to introduce additional pessimism by specifying a higher quantile.

6 Example

Consider the task system in Table 1 on a four-core processor. This example demonstrates our main contribution: each task has a worst-case execution time greater than its period, so it would be considered unschedulable by [1, 4]. However, since the total expected utilization is 3.2, which is less than 4.0, and each task's expected utilization is less than one, τ is stable by

Task	p_i	$\bar{e}_i$	e_i	σ_i^2	$\bar{u}_i$	$\hat{u}_i$
τ_1	4	3	25	1	0.75	0.86
τ_2	4	3	20	1	0.75	0.86
τ_3	5	3	30	4	0.60	0.96
τ_4	5	3	20	1	0.60	0.69
τ_5	8	2	15	1	0.25	0.31
τ_6	20	3	35	2	0.15	0.20
τ_7	20	2	25	1	0.10	0.12

Table 1: Example Task System τ.

$$
\begin{array}{lll}
\max & \zeta & \\
\text{s.t.} & 4\hat{u}_1 - 0.5\zeta \geq 3 & 0.75 \leq \hat{u}_1 \leq 1 \\
& 4\hat{u}_2 - 0.5\zeta \geq 3 & 0.75 \leq \hat{u}_2 \leq 1 \\
& 5\hat{u}_3 - 2\zeta \geq 3 & 0.60 \leq \hat{u}_3 \leq 1 \\
& 5\hat{u}_4 - 0.5\zeta \geq 3 & 0.60 \leq \hat{u}_4 \leq 1 \\
& 8\hat{u}_5 - 0.5\zeta \geq 2 & 0.25 \leq \hat{u}_5 \leq 1 \\
& 20\hat{u}_6 - \zeta \geq 3 & 0.15 \leq \hat{u}_6 \leq 1 \\
& 20\hat{u}_7 - 0.5\zeta \geq 2 & 0.10 \leq \hat{u}_7 \leq 1 \\
& \sum_{i=1}^{7} \hat{u}_i \leq 4 &
\end{array}
$$

Figure 4: Linear Program for example.

our definition. Using the means and variances given in Table 1, we can apply the result of Corollary 10.

To find ψ, we set up the linear program (LP), which is written out in Fig. 4. A standard LP solver will give us the solution in Table 1. The value of ζ^* is 0.90, so ψ equals 1.11. An upper bound on $\eta_{i,j}$ is the sum of the three most expensive worst-case execution times, which is 90. υ is the sum of the three most expensive values of $\hat{u}_i$, which are 0.96, 0.86, and 0.86, so υ equals 2.68.

We conclude that the expected (mean) tardiness of any job $\tau_{i,j}$, taken over all possible instances of this task system when scheduled on four processors, has the following upper bound: $\mathsf{E}\left(T_{i,j}^{\mathsf{GEDF}}\right) \leq 1.11\hat{u}_i + 71.55 + e_i$. Once we have this upper bound for the mean of $T_{i,j}^{\mathsf{GEDF}}$, we can generate upper bounds for the q-quantile by dividing by $1 - q$. For example, if we want an upper bound on the 90th percentile for tardiness in GEDF, we would divide $\mathsf{E}\left(T_{i,j}^{\mathsf{GEDF}}\right)$ by 0.1.

This example demonstrates the practical value of our result—using the method of [1], this system is unschedulable, and could not be made schedulable by removing tasks or adding processors, since each task has a worst-case execution time that exceeds its period. Furthermore, even if jobs were allowed to execute in parallel, the worst-case utilization of this system is over 26, so we would need at least 27 processors.

7 Concluding Remarks

We presented a probabilistic tardiness-bound derivation for GEDF that is a generalization of the result of [1]. If we use a deterministic model where worst-case execution times are required for every job, then $\bar{e}_i = e_i, \forall i$, $\sigma_i^2 = 0, \forall i$, and then choosing $\psi = 0$, $\hat{u}_i = e_i/p_i, \forall i$ satisfies (34); in other words, there is no PS-induced tardiness. Since the system is deterministic, tardiness and expected tardiness are equivalent, resulting in the tardiness bound for any job of τ_k of $\frac{\eta}{m-\upsilon} + e_k$, which almost matches the tardiness bound given in [1], which is $\frac{\eta - e_{\min}}{m-\upsilon} + e_k$, where $e_{\min}$ is the smallest worst-case execution time over all tasks.

In future work, we would like to expand this result to allow for different tasks to have different (specified) tardiness bounds. This would be useful from a system design perspective; given a list of desired expected tardiness bounds for each task, we would like, for example, to be able to determine the minimum number of processors needed to schedule the task system.

In addition, both theoretical and empirical work are needed to determine the impact of the assumption of independence of execution times, which was made for purposes of tractability. For example, a first step may be to introduce some autocorrelation among execution times of sequential jobs of the same task.

Another area of future work is to determine how other deterministic results about soft real-time scheduling, such as established tardiness bounds for other global scheduling algorithms [4], can be generalized for the case of stochastic execution times. We conjecture that the proof in this paper could be extended to any scheduling algorithm with window-constrained priorities, as defined in [4].

References

[1] U. C. Devi and J. H. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, 2005.

[2] B. Doytchinov, J. Lehoczky, and S. Shreve. Real-time queues in heavy traffic with earliest-deadline-first queue discipline. *The Annals of Applied Probability*, 11(2), 2001.

[3] D. P. Heyman and M. J. Sobel. *Stochastic Models in Operations Research*, volume 1. McGraw-Hill, 1982.

[4] H. Leontyev and J. H. Anderson. Generalized tardiness bounds for global multiprocessor scheduling. In *Proceedings of the 28th IEEE Real-Time Systems Symposium*, 2007.

[5] D. V. Lindley. The theory of queues with a single server. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48(2), 1952.

[6] S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, 2009.

[7] S. Ross. *A first course in probability*. Prentice Hall, 6 edition, 2002.

[8] J. A. Van Mieghem. Due-date scheduling: Asymptotic optimality of generalized longest queue and generalized largest delay rules. *Operations Research*, 2003.

[9] P. Zijlstra. Deadline scheduling in Linux and why it hasn't happened yet. In *Eleventh Real Time Linux Workshop*, 2009.