Analysis of Long-term Average Behaviors of Probabilistic Task Systems

Yifan Cai
University of Pennsylvania
USA
caiyifan@seas.upenn.edu

Linh Thi Xuan Phan
University of Pennsylvania / Roblox
USA
linhphan@cis.upenn.edu

P.S. Thiagarajan
University of North Carolina at Chapel Hill
USA
Chennai Mathematical Institute
India
thiagu@cs.unc.edu

Abstract

We present a Markov chain-based framework for studying the long-term average behaviors of periodic real-time task systems in which the tasks have stochastically varying computation times. In sharp contrast to previous work, we construct our Markov chains w.r.t. to a unit of time that is *not* required to be the hyperperiod of the task system. Our chains have an important property called irreducibility, and this secures the mathematical basis for a simple sampling procedure for estimating the long-term averages of interest. This is significant because for task systems of practical interest, it will be computationally infeasible to use hyperperiods to determine the required expected values. Our experimental results show that our method can be used to analyze long-term average behaviors – such as deadline misses and weakly-hard constraint violations – with high accuracy, and that it scales well to large systems (with up to 1000 tasks). We further demonstrate its practical utility using a case study of a rover control system.

CCS Concepts

• **Computer systems organization → Real-time systems**; • **Theory of computation → Random walks and Markov chains**.

Keywords

Real-time systems, Markov chain, Ergodic theorem, weakly-hard constraints, sampling

ACM Reference Format:
Yifan Cai, Linh Thi Xuan Phan, and P.S. Thiagarajan. 2024. Analysis of Long-term Average Behaviors of Probabilistic Task Systems. In *The 32nd International Conference on Real-Time Networks and Systems (RTNS 2024), November 07–08, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696355.3696365

1 Introduction

Due to the continued increase in hardware/software complexity, it has become extremely challenging to obtain tight worst-case execution time bounds for modern real-time systems. As a result, probabilistic real-time task models, where one or more of the timing parameters such as task execution times are modeled by random variables, have been used for analyzing real-time systems [10].

In the probabilistic setting, existing analysis techniques focus primarily on *worst-case* properties, such as the worst-case deadline miss probabilities. For many systems, however, it is also important to consider *long-term average* behaviors. For instance, soft real-time and latency-sensitive applications, such as real-time streaming video/audio applications, typically require a certain quality of service (QoS) defined based on long-term average performance metrics. Likewise, in mixed-criticality real-time systems, for a non-critical or low-criticality task, it suffices to ensure its long-term average deadline miss rate is within a certain threshold. In the case of feedback control systems too, control tasks (such as adaptive cruise control and collision avoidance) can be designed to tolerate *some* deadline miss patterns without sacrificing stability and quality of control [30]. For such control tasks, we only need to ensure that the long-term average deadline miss rates are within the tolerance levels specified by the control designers. A recent trend [9, 19, 20, 24, 35] in this setting is to soften the deadline miss requirements via *weakly-hard* constraints [6, 18]. For instance, the weakly-hard constraint (m, k) demands that in any k consecutive invocations of a task, there must be at least m deadline hits. An important variant is: In any k consecutive invocations of a task, there cannot be m *consecutive* deadline misses. These kinds of guarantees often require much fewer resources than worst-case guarantees, while still yielding the desired performance. However, new types of analysis are required here since existing techniques are primarily designed for analyzing worst-case probabilities.

Motivated by these considerations, we present a technique for estimating the long-term averages of properties associated with probabilistic task systems. In particular, we focus on periodic task systems in which the computation times of each task can vary stochastically according to an associated probability distribution over a finite set of computation times. Thus, as a first step, we treat the computation times as independent random variables. As we observe in connection with our case study (Sec. 5), this can be a useful working hypothesis. This class of systems has been extensively studied in the literature from various angles as we describe in Sec. 6. Here we sketch our main ideas and results.

Overview. Our first goal is to formalize the notion of long-term averages. Intuitively, it is the average number of, say, the number of deadline misses of a task divided by the total number

RTNS 2024, November 07–08, 2024, Porto, Portugal

ACM ISBN 979-8-4007-1724-6/24/11
https://doi.org/10.1145/3696355.3696365

of deadlines in an unbounded time window. The question then is what does an unbounded time window mean? The standard abstraction for this idea is an infinite time window. Accordingly, we define the infinite execution sequences of a probabilistic periodic task system. Then one might define the long-term average as the limit of averages of finite prefixes of increasing lengths of infinite execution sequences. The catch here is one must ensure such a limit always exists. We achieve this by establishing that the behavior of the task systems can be modeled as a finite-state Markov chain and that it captures the infinite execution sequences of the task system. We further show that this Markov chain has a standard property called irreducibility. Informally, it says that the state space of the Markov chain is strongly connected. As a result, we can appeal to a fundamental result called the ergodic theorem to guarantee that sequences of averages of finite prefixes will converge to a definite value called the expected value. As an important byproduct, we obtain a simple sampling procedure for estimating the long-term averages of properties such as deadline misses *and* weakly-hard constraints.

A key aspect of our Markov chain construction is that, in contrast to most prior research on Markov chain-based analysis of periodic task systems, it is *not* based on hyperperiods (i.e., the least common multiple of the periods of all the tasks). Instead, it is based on a user-supplied notion of unit interval. This unit interval can be much smaller than the hyperperiod, which can be enormous (e.g., 10^7 in our rover control case study). Consequently, the sampling procedure can be much more efficient (See Fig. 5 in Sec. 5 and the accompanying remarks).

The finite-state Markov chains that arise from our work constitute a fundamental model of stochastic processes with a rich theory and a wide variety of applications. Hence, using them to model the infinite behaviors of probabilistic task systems opens up the possibility of developing many other analysis methods. In particular, probabilistic model checking techniques [21] become accessible.

Contributions. In summary, the paper makes the following contributions: (1) Introducing a finite-state Markov chain model that represents the infinite behaviors of periodic probabilistic task systems. (2) Establishing the property called irreducibility for the Markov chain model. This provides a mathematical basis for estimating long-term averages of properties such as deadline misses and violations of weakly-hard constraints using lightweight sampling procedures. (3) Building this theory *without* basing it on the hyperperiods of the task systems. To the best of our knowledge, this is the first time this has been achieved. Another novel aspect of our method is that it can smoothly handle a rich language of weakly-hard constraints. (4) Presenting an experimental evaluation demonstrating that our sampling-based method has high accuracy for both synthetic workloads and a real-world rover control system. Our evaluation also shows that the method scales well.

Organization of the paper. In Sec. 2, we formulate the system model, define the basic notion of a configuration, which captures the state of the system at a given time point, and define the infinite execution sequences of the system. In Sec. 3, we introduce unit intervals and Markov chain preliminaries. Sec. 4 is the technical core of the paper. First, we define a one-step transition relation between configurations and then lift it to a multi-step relation. We then establish the Markovian properties of these transition relations. This leads to an infinite state Markov chain representing the set of execution sequences of the task system. Each state of this chain will represent a unit stretch of the behavior of the system. Next, by defining a natural equivalence relation over the states of this chain, we construct a finite state Markov chain, the key object in our study. We then show that this chain is irreducible and consequently its so-called stationary distribution – explained in Section 3 – captures the long-term averages of many properties of interest. Furthermore, it provides the formal basis for our simple sampling procedure for estimating these averages. These results are established assuming a preemptive static priority policy. However, our technique can be extended in a straightforward manner to dynamic priority schemes such as EDF. We then sketch how our construction can be used to analyze weakly-hard constraints. In Sec. 5, we present our experimental results. We discuss related work in Sec. 6 and conclude with future directions in Sec. 7.

2 The System Model

The system consists of a set of periodic tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$ running on a single processor. Each task τ_i has a period T_i, a deadline $D_i \leq T_i$, and an initial offset $\varphi_i \geq 0$ (i.e., jobs of τ_i will be released at time points $\varphi_i + kT_i$ for $k \geq 0$). We consider a preemptive static priority scheduling scheme. As mentioned earlier, our method can be easily extended to a non-preemptive or dynamic priority scheme such as EDF. For convenience, we assume $D_i = T_i$ and $\varphi_i = 0$ for $1 \leq i \leq n$. Our results will easily extend to the case $D_i < T_i$. The case $D_i > T_i$ doesn't come into play since, as in many previous studies, we adopt the policy that a job is killed as soon it misses its deadline.

A key feature of our model is the computation times of a job can vary stochastically. To capture this, we assume a finite set of computation times $C_i = \{c_1^i, c_2^i, \ldots, c_{K_i}^i\}$ where $0 < c_1^i < c_2^i < \ldots < c_{K_i}^i$ and a probability distribution $Pr_i : C_i \to (0, 1]$ for each τ_i. As is common in prior work, we assume that for each job $\tau_{i,j}$ released by a task τ_i during an execution, its execution time is chosen independently (of previous history and jobs of other tasks) from C_i according to Pr_i. In future work, we plan to explore settings in which this assumption is relaxed.

Formally, we focus on $\mathcal{S} = \{(T_i, D_i, C_i, Pr_i)\}_{1 \leq i \leq n}$, a periodic task system, with $D_i = T_i$ for $1 \leq i \leq n$. Following [7], we assume that the time domain $\mathbb{T}$ is discretized w.r.t. to a chosen micro time unit γ (e.g., the processor cycle). Thus, $\mathbb{T} = \{k \cdot \gamma \mid k \geq 0\}$ with k ranging over $\mathbb{N}_0$, the set of non-negative integers. All the temporal quantities encountered, such as T_i and the members of C_i, are assumed to be integral multiples of γ and hence will be viewed as integers with the factor γ almost always suppressed.

2.1 Configurations

We begin with $J_i^\infty = \{\tau_{i,k} \mid k \geq 0\}$, the infinite set of jobs released by τ_i along an infinite execution sequence, with $\tau_{i,k}$ being released at time kT_i. This leads to $J^\infty = \bigcup_i J_i^\infty$. As done here, we will often abbreviate the index set $1 \leq i \leq n$ as just i. In addition, we will say that a job belongs to the task τ_i if it is a member of J_i^∞. The discretized time points along an infinite run will be denoted as non-negative integers with t representing the time point $t \cdot \gamma$.

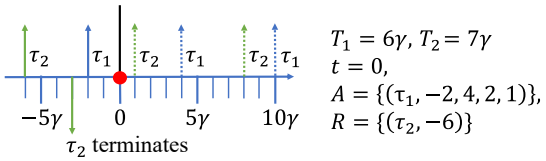

Figure 1: An example of configurations at time $t = 0$.

Configurations. A configuration describes the state of the system at time point t. Formally, it is a triple $\mathbf{u} = (t, A, R)$, where t is a non-negative integer, and A and R are defined as follows:

(1) A records information concerning the set of *active* jobs that were released before or at t which have neither missed their deadlines nor have finished computing at t. Each member of A will be of the form $(\tau_i, t_i, c_i, d_i, st_i)$, where:
- τ_i is the task that this job belongs to.
- t_i is the time at which this job was released. Since the job is active at time t, we require $t_i \le t < t_i + T_i$. We note that for some non-negative integer l, we will have $t_i = lT_i$.
- $c_i \in C_i$ is the computation time assigned to the job (with probability $Pr_i(c_i)$) when it is released.
- d_i is the remaining computation time of this job at t, where $0 < d_i \le c_i$.
- $st_i \in \{0, 1\}$ is the status of this job during $[t, t+1)$, where 1 denotes that the job is currently running, and 0 denotes that it is currently waiting. We require that at most one job in A has status 1, and this job (if A is non-empty) has the highest priority among all the active jobs in A.

(2) R is the set of tasks that are at *rest* at t. Each member of R will be a pair (τ_i, t_i) where t_i is the time point at which the last *completed* job belonging to τ_i was released but whose next instance has not been released up to and including t. Thus, we require that there exists a non-negative integer l such that $t_i = l \cdot T_i$ and $t_i < t < t_i + T_i$.

We also require that each task must appear in A or R, but not both. More precisely, let $\mathcal{T}_A = \{\tau_i \mid a \in A \text{ and } a(1) = \tau_i\}$ and $\mathcal{T}_R = \{\tau_i \mid r \in R \text{ and } r(1) = \tau_i\}$. We require $\mathcal{T}_A \cap \mathcal{T}_R = \emptyset$ and $\mathcal{T}_A \cup \mathcal{T}_R = \{\tau_i\}_i$. We use $a(1)$ to denote the first component of the 5-tuple a, and $r(1)$ to denote the first component of the pair r.

Example. Fig. 1 shows an example of a configuration at $t = 0$ for a task system with two tasks τ_1 and τ_2 with periods $T_1 = 6$ and $T_2 = 7$, respectively. At $t = 0$, since the last job of τ_2 released at $t_2 = -6$ has finished at -3, and its new job hasn't been released, only τ_1 has an active job. This job was released at $t_1 = -2$ with computation time $c_1 = 4$. It has finished 2 unit time of computation and is running at $t = 0$ and thus $d_1 = 2$ and $st_1 = 1$.

We note that the last component of each member of A is uniquely determined by the static priority scheme we are assuming. Hence, we will not spell out the last component of the elements of A. In other words, from now on, a configuration will be represented as a triple (t, A, R) with each member of A, a quadruple (τ_i, t_i, c_i, d_i). We let $\mathcal{H}$ denote the set of all configurations.

Transitions. We next specify how the system transitions from one configuration to the next one.

Definition 1. *Let $\mathbf{u} = (t, A, R)$ and $\mathbf{v} = (t', A', R')$ be two configurations. Then, $\mathbf{u} \longrightarrow \mathbf{v}$ iff the following conditions hold:*

C1. $t' = t + 1$.

C2. $(\tau_j, t_j, c_j, d_j) \in A'$ *iff:*

C2.1. $t_j = t + 1 = kT_j$ *for some* $k \in \mathbb{N}_0$ *and* $c_j \in C_j$ *and* $d_j = c_j$, *or*

C2.2. $t + 1 = kT_j$ *for no* $k \in \mathbb{N}_0$ *and there exists* $(\tau_j, t_j, c_j, d) \in A$ *with* $d - st > 0$ *and* $d_j = d - st$.

C3. $(\tau_j, t_j) \in R'$ *iff:*

C3.1. $(\tau_j, t_j) \in R$ *and* $t + 1 = kT_j$ *for no* $k \in \mathbb{N}_0$, *or*

C3.2. *there exists* $(\tau_j, t_j, c, 1) \in A$ *such that: (i)* τ_j *has the highest priority among the active jobs at* t *and (ii)* $t + 1 = kT_j$ *for no* $k \in \mathbb{N}_0$

(C1) states that $\mathbf{v}$ is a configuration that holds at $t + 1$. (C2.1) indicates that a new job of τ_j is released at $t + 1$, with computation time c_j and the remaining computation time also c_j. On the other hand, (C2.2) indicates that an existing job of τ_j is active at t and remains active at $t + 1$ (since at $t + 1$, this job still has at least 1 unit of computation remaining and has not reached its deadline yet), and its remaining execution time at $t + 1$ is the same (if $st_i = 0$) or one less (if $st_i = 1$) than at t.

(C3.1) states that τ_j is at rest at t and will remain so at $t + 1$ since its next instance will be released at a time point later than $t + 1$. As for (C3.2), τ_j is active and executing at time t, and has only one computation unit left. Hence, it would be at rest at $t + 1$ since its next instance will be released only at a later time point. The configuration $\mathbf{v}$ is a *successor* of the configuration $\mathbf{u}$ iff $\mathbf{u} \longrightarrow \mathbf{v}$.

Finally, (t, A, R) is an *initial* configuration if and only if $t = 0$, $A = \{(\tau_i, 0, c_i, c_i) \mid c_i \in C_i\}$, and $R = \emptyset$. We let $\mathcal{H}_{in}$ denote the set of initial configurations of $\mathcal{S}$.

Traces and execution sequences.

(1) A finite *trace* of $\mathcal{S}$ is a sequence of configurations $\rho = \mathbf{u}_0\mathbf{u}_1 \cdots \mathbf{u}_k$ such that $\mathbf{u}_l \longrightarrow \mathbf{u}_{l+1}$ for $0 \le l < k$. The trace ρ is said to start from $\mathbf{u}_0$.
(2) An *infinite* trace is an infinite sequence of configurations such that its every finite prefix is a finite trace.
(3) A finite *execution* sequence is a finite trace that starts from an initial configuration.
(4) An infinite execution sequence is defined in an obvious way.

We denote by $B_{\mathcal{S}}$ the set of infinite execution sequences of $\mathcal{S}$ and view it as the behavior of $\mathcal{S}$.

We conclude this section with a useful result about the successors of a configuration: every finite execution sequence can be extended to a longer one, and hence the behavior of $\mathcal{S}$ is assured to be non-empty. This result will also form the basis of the Markov chain constructions developed in the next section.

To start with, let $\mathbf{u} = (t, A, R)$ be a configuration. Then $X_{\mathbf{u}}$ captures information about the set of jobs released at $t + 1$. It is the subset of tasks defined as: $\tau_i \in X_{\mathbf{u}}$ iff there exists $(\tau_i, t_i, c_i, d_i) \in A$ or $(\tau_i, t_i) \in R$ such that $t_i + T_i = t + 1$.

Proposition 2.1. *Let $\mathbf{u} = (t, A, R)$ be a configuration.*

(1) If $X_{\mathbf{u}} = \emptyset$, then $\mathbf{u}$ has exactly one successor configuration.
(2) Suppose $X_{\mathbf{u}} = \{\tau_{i_1}, \tau_{i_2}, \cdots, \tau_{i_k}\}$. Then, for each $\mathbf{c} \in C_{i_1} \times C_{i_2} \times \cdots \times C_{i_k}$, there exists a unique successor configuration $\mathbf{v_c} = (t + 1, A_{\mathbf{c}}, R_{\mathbf{c}})$ of $\mathbf{u}$ satisfying: For $1 \le l \le k$, $(\tau_{i_l}, t + 1, \mathbf{c}(l), \mathbf{c}(l)) \in A_{\mathbf{c}}$.

Proof. Suppose $X_{\mathbf{u}} = \emptyset$. Consider the triple $\mathbf{v} = (t + 1, A', R')$ given by:

(1) Suppose $(\tau_i, t_i, c_i, d_i) \in A$ and $st_i = 1$. If $d_i = 1$ then $(\tau_i, t_i) \in R'$. If $d_i > 1$ then $(\tau_i, t_i, c_i, d_i - 1) \in A'$.
(2) If $(\tau_i, t_i, c_i, d_i) \in A$ and $st_i = 0$ then $(\tau_i, t_i, c_i, d_i) \in A'$.
(3) If $(\tau_i, t_i) \in R$ then $(\tau_i, t_i) \in R'$.

From the definition of a configuration and the fact that $X_{\mathbf{u}} = \emptyset$, it follows easily that $\mathbf{v}$ is a configuration, is uniquely induced by $\mathbf{u}$, and is a successor configuration of $\mathbf{u}$. We note that $X_{\mathbf{u}} = \emptyset$ also implies that whether $st_i = 0$ or $st_i = 1$ during $[t, t+1)$, there can be no deadline miss occurring at $t+1$.

Next, assume that $X_{\mathbf{u}} = \{\tau_{i_1}, \tau_{i_2}, \cdots, \tau_{i_k}\}$. For each $\mathbf{c} \in C_{i_1} \times C_{i_2} \times \cdots \times C_{i_k}$, let $(t+1, A_{\mathbf{c}}, R_{\mathbf{c}})$ be given by:

(1) For $1 \leq l \leq k$, $(\tau_{i_l}, t+1, \mathbf{c}(l), \mathbf{c}(l)) \in A'$.
(2) Suppose $\tau_i \notin X_{\mathbf{u}}$.
   - If $(\tau_i, t_i, c_i, d_i) \in A$, $d_i = 1$ and $st_i = 1$ then $(\tau_i, t_i) \in R'$.
   - If $(\tau_i, t_i, c_i, d_i) \in A$ and $d_i > 1$ or $st_i = 0$ then $(\tau_i, t_i, c_i, d_i) \in A'$.
   - If $(\tau_i, t_i) \in R$ then $(\tau_i, t_i) \in R'$.

From the definitions, it follows that $(t+1, A_{\mathbf{c}}, R_{\mathbf{c}})$ is a configuration, is uniquely determined by $\mathbf{u}$, and is a successor configuration of $\mathbf{u}$. This proves the second part of the proposition. □

3 Unit Intervals and Markov Chains

3.1 Unit intervals

We will construct a finite-state Markov chain relative to a given unit of time Δ. We require Δ to be an integral multiple of time but again with the factor γ suppressed. A state of the Markov chain will be a trace of length Δ. At the end of the current Δ interval, the chain will transition to the next trace of length Δ. We assume that Δ has been fixed based on some pragmatic considerations. Some possible choices for Δ are $\max(\{T_i\})$, $K \cdot \max(\{T_i\})$ for some small positive integer K, or the lcm of the periods of a small subset of $\{T_i\}$. A large Δ (e.g., $\Delta = lcm(\{T_i\}_i)$ = hyperperiod) will cause a state to carry excessive information, while a small Δ (e.g., $\Delta = \gamma$) will require longer sequences of paths to be sampled before instances of the properties of interest appear in sufficient numbers. The impact of the choice of Δ is evaluated more systematically in Sec. 5.

3.2 Markov chain preliminaries

Our sampling-based analysis technique will be based on *finite* state Markov chains. However, we will first represent the behavior of $\mathcal{S}$ with an infinite-state Markov chain and then quotient it into a finite-state chain. Hence we begin with:

Definition 2. *A Markov chain is a structure $M = (S, S_{in}, P)$ where:*

(1) S is a non-empty countable set of states.
(2) $S_{in} \subseteq S$ is a set of initial *states.*
(3) $P : S \times S \to [0, 1]$ is the probabilistic transition function satisfying: for every $s \in S$, $\sum_{s' \in S} P(s, s') = 1$.

Let $M = (S, S_{in}, P)$ be a *finite-state* Markov chain with $S = \{s_1, s_2, \cdots, s_m\}$. Then M can be represented as the edge labeled directed graph $G_M = (S, E)$ where $E = \{(s_i, s_j) \mid P(s_i, s_j) > 0\}$. If $(s_i, s_j) \in E$ then $P(s_i, s_j)$ is the label on this edge. A fundamental property of Markov chains is *irreducibility*. M is said to be irreducible iff G_M is a strongly connected graph. This property implies that M has a unique *stationary* distribution over its set of states. This is best brought out in an algebraic setting.

The transition matrix of M, also denoted as M by abuse of notation, is the $|S| \times |S|$ matrix satisfying: $M(i, j) = P(s_i, s_j)$. This matrix M is viewed as a transformer of probability distributions over the states of the chain. If μ is a distribution over S, then M will transform it into a new distribution μ' in one step. This one-step transformation is represented as the matrix multiplication $\mu \cdot M = \mu'$ where a distribution over S is represented as a $1 \times m$ row vector.

π is a *stationary* distribution of M iff $\pi \cdot M = \pi$. The basic property of an irreducible Markov chain is that it has a *unique* stationary distribution [28]. The key feature of the stationary distribution is that it captures the "long-term average values" of quantities associated with the dynamics of the chain. Suppose π is the stationary distribution of the irreducible finite-state Markov chain M whose set of states is S. Then intuitively, in the long run, if we guess the current state of M to be s, then this guess will be correct with probability $\pi(s)$. Equally important, the average number of times that s will appear in any randomly sampled (i.e. sampled according to the transition probabilities) finite path, will, in the limit, converge to $\pi(s)$. This follows from the ergodic theorem [28] for irreducible Markov chains which we state next.

A finite path of M is a sequence $\xi = s_0 s_1 \cdots s_m$ such that $P(s_l, s_{l+1}) > 0$ for $0 \leq l < m$. We define $|\xi| = m$ where $|\xi|$ denotes the length of ξ and let $\xi(l) = s_l$ for $0 \leq l \leq m$. If ξ and ξ' are two finite paths then $\xi \prec \xi'$ denotes that ξ is a prefix of ξ' and $|\xi| < |\xi'|$. That is, ξ is a strict prefix of ξ'.

Next, we define a random variable over S to be a function $f : S \to \mathbb{R}$ where $\mathbb{R}$ is the set of reals. Suppose M with S as its set of states is a finite-state irreducible Markov chain, π is its stationary distribution and f is a random variable over S. Then, $\widehat{f} = \sum_{s \in S} \pi(s) \cdot f(s)$. Often $\widehat{f}$ is written as $\mathbb{E}(f)$ and is called the *expectation* of f. The following classic result, often called the ergodic theorem, provides the mathematical basis for our sampling procedure.

Theorem 1 (Theorem 1.10.2 in [28]). *Let $M = (S, S_{in}, P)$ be an irreducible finite-state Markov chain and $f : S \to \mathbb{R}$ be a random variable. Suppose $\{\xi_k\}_{k \geq 1}$ is an infinite sequence of finite paths with $|\xi_k| = k$ and $\xi_k \prec \xi_{k+1}$ for $k \geq 1$. Then,*

$$\lim_{k \to \infty} \frac{1}{k} \sum_{j=1}^{k} f(\xi_k(j)) = \widehat{f} \text{ (with probability 1).}$$

To be precise, the theorem as stated above is a specialized version of Theorem 1.10.2 (ergodic theorem) in [28]. In the original version, f is required to be a bounded function. This is assured in our case since S is a finite set. The theorem also requires M to have a property called positive recurrence. Again, this is guaranteed in our case since M is an irreducible finite-state chain [28]. Finally, the proviso "with probability 1" is a measure-theoretic technicality. Loosely speaking, it says that under the usual probability measure defined over collections of infinite paths of M, the set of infinite paths for which the claimed convergence property holds will have probability measure 1. Note that the sequence of finite paths $\{\xi_k\}_k$ with $\xi_k \prec \xi_{k+1}$ uniquely defines an infinite path of the Markov chain. Thus, the sequence of averages defined in the ergodic theorem are averages taken over the finite prefixes of increasing lengths of a randomly sampled infinite path of the chain; and this sequence will almost certainly converge to the expectation of the random variable.

An important consequence of the ergodic theorem is that this convergence property does not depend on the starting state of the randomly sampled path.

4 The Main Results

Through this section, we fix a unit interval $\Delta \geq 1$. We begin by deriving two important probabilistic transition relations.

4.1 Two probabilistic transitions

To start with we associate the set $\alpha_{\mathbf{u}}$ and the probability value $pr(\mathbf{u})$ with the configuration $\mathbf{u}$. They are defined as follows.

Let $\mathbf{u} = (t, A, R)$. Then, $\alpha_{\mathbf{u}} = \{(\tau_i, c_i) \mid (\tau_i, t, c_i, c_i) \in A\}$. Thus, $\alpha_{\mathbf{u}}$ records the stochastically assigned computation times to the jobs released at t. If $\alpha_{\mathbf{u}} = \{(\tau_{i_1}, c_{i_1}), (\tau_{i_2}, c_{i_2}) \cdots, (\tau_{i_k}, c_{i_k})\}$, then $pr(\mathbf{u}) = \prod_{1 \leq l \leq k} Pr_{i_l}(c_{i_l})$. Hence, $pr(\mathbf{u})$ is the product of the probabilities with which the computation times of the jobs released at t are chosen. If $\alpha_{\mathbf{u}} = \emptyset$, then $pr(\mathbf{u}) = 1$ by convention.

Definition 3. *Let $\mathbf{u} = (t, A, R)$ and $\mathbf{v} = (t', A', R')$ be two configurations and $p \in (0, 1]$. Then $\mathbf{u} \xrightarrow[1]{p} \mathbf{v}$ iff $\mathbf{v}$ is a successor configuration of $\mathbf{u}$ (and hence $t' = t + 1$) and $pr(\mathbf{v}) = p$.*

Recall that $\mathcal{H}$ is the set of configurations of $\mathcal{S}$. It will be convenient to represent this probabilistic transition relation as the function $P^1 : \mathcal{H} \times \mathcal{H} \rightarrow [0, 1]$ given by $P^1(\mathbf{u}, \mathbf{v}) = p$, if $\mathbf{u} \xrightarrow[1]{p} \mathbf{v}$. Otherwise $P^1(\mathbf{u}, \mathbf{v}) = 0$. It will also be convenient to denote by $succ^1(\mathbf{u})$ the set of configurations $\{\mathbf{v} \mid P^1(\mathbf{u}, \mathbf{v}) > 0\}$. The next observation is key to our Markov chain constructions.

Lemma 2. *Let $\mathbf{u}$ be a configuration. Then $\sum_{\mathbf{v} \in succ^1(\mathbf{u})} P^1(\mathbf{u}, \mathbf{v}) = 1$*

Proof. If $X_{\mathbf{u}} = \emptyset$ then there exists a unique $\mathbf{v}$ such that $succ^1(\mathbf{u}) = \{\mathbf{v}\}$. Furthermore, we are assured that $\alpha_{\mathbf{v}} = \emptyset$ and hence $pr(\mathbf{v}) = 1$. This leads to $\sum_{\mathbf{w} \in succ^1(\mathbf{u})} P^1(\mathbf{u}, \mathbf{w}) = 1$.

So assume that $X_{\mathbf{u}} = \{\tau_{i_1}, \tau_{i_2}, \cdots, \tau_{i_k}\}$. Let $\mathbf{C} = C_{i_1} \times C_{i_2} \times \cdots \times C_{i_k}$. Then according to Prop.2.1, for each $\mathbf{c} \in \mathbf{C}$, there exists a unique successor configuration $\mathbf{v_c} = (t + 1, A_{\mathbf{c}}, R_{\mathbf{c}})$ of $\mathbf{u}$ satisfying: for $1 \leq l \leq k$, $(\tau_{i_l}, t + 1, \mathbf{c}(l), \mathbf{c}(l)) \in A_{\mathbf{c}}$. This implies that $\alpha_{\mathbf{v_c}} = \{(\tau_{i_l}, \mathbf{c}(l)) \mid 1 \leq l \leq k\}$ for each $\mathbf{c} \in \mathbf{C}$, which in turn implies that $pr(\mathbf{v_c}) = \prod_{1 \leq l \leq k} Pr_{i_l}(\mathbf{c}(l))$. Furthermore, $\sum_{\mathbf{v} \in succ^1(\mathbf{u})} P^1(\mathbf{u}, \mathbf{v}) = \sum_{\mathbf{c} \in \mathbf{C}} p_{\mathbf{c}}$ where $p_{\mathbf{c}} = pr(\mathbf{v_c})$ for $\mathbf{c} \in \mathbf{C}$.

We now proceed by induction on k. If $k = 1$ then $\mathbf{C} = C_{i_1}$. Clearly, $pr(\alpha_{\mathbf{v}_c}) = Pr_{i_1}(c)$ for $c \in C_{i_1}$. But then $\sum_{c \in C_{i_1}} Pr_{i_1}(c) = 1$.

So assume that $k > 1$. As before, let $p_{\mathbf{c}} = pr(\mathbf{v_c})$ for $\mathbf{c} \in \mathbf{C}$. Then $\sum_{\mathbf{c} \in \mathbf{C}} p_{\mathbf{c}} = \sum_{\mathbf{c}' \in \mathbf{C}'} p_{\mathbf{c}'} \cdot (\sum_{c \in C_{i_k}} Pr_{i_k}(c))$ where $\mathbf{C}' = C_{i_1} \times C_{i_2} \times \cdots \times C_{i_{k-1}}$. From the proof of the basis step, it follows that $\sum_{c \in \mathbf{C}_{i_k}} Pr_{i_k}(c) = 1$. Thus $\sum_{\mathbf{c} \in \mathbf{C}} p_{\mathbf{c}} = \sum_{\mathbf{c}' \in \mathbf{C}'} p_{\mathbf{c}'}$. The result now follows from the induction hypothesis. □

We next extend $\xrightarrow[1]{p}$ to traces of length Δ. This is motivated by the fact that the states of the infinite-state chain we wish to first construct will be traces of length Δ. To define this extension, we first lift pr to a non-null sequence of configurations $\mathbf{x} = \mathbf{u}_0\mathbf{u}_1 \cdots \mathbf{u}_m$ via: $pr(\mathbf{x}) = \prod_{0 \leq l \leq m} pr(\mathbf{u}_l)$. Next, in what follows, TR_Δ will denote the set of traces of length Δ. Finally, if $\mathbf{s} \in TR_\Delta$ with $\mathbf{s} = \mathbf{u}_0\mathbf{u}_1 \cdots \mathbf{u}_{\Delta-1}$, then where convenient, we shall view $\mathbf{s}$ to be the map $\mathbf{s} : \{0, 1, \Delta - 1\} \rightarrow \mathcal{H}$ with $\mathbf{s}(k) = \mathbf{u}_k$ for $0 \leq k < \Delta$. Now let $\mathbf{s}, \mathbf{s}' \in TR_\Delta$. Then $\mathbf{s} \xrightarrow[\Delta]{p} \mathbf{s}'$ iff (i) $p = pr(\mathbf{s}')$ and (ii) $\mathbf{s}'(0)$ is a successor configuration of $\mathbf{s}(\Delta - 1)$. We now extend the function P^1 to P^Δ via: $P^\Delta(\mathbf{s}, \mathbf{s}') = p$ if $\mathbf{s} \xrightarrow[\Delta]{p} \mathbf{s}'$; otherwise, $P^\Delta(\mathbf{s}, \mathbf{s}') = 0$. As before, it will also be convenient to define $succ^\Delta(\mathbf{s}) = \{\mathbf{s}' \mid P^\Delta(\mathbf{s}, \mathbf{s}') > 0\}$. This leads to the next lemma.

Lemma 3. *Let $\mathbf{s} \in TR_\Delta$. Then $\sum_{\mathbf{s}' \in succ^\Delta(\mathbf{s})} P^\Delta(\mathbf{s}, \mathbf{s}') = 1$*

The proof follows easily by induction on Δ with Lemma 2 serving as the proof of the basis step.

4.2 The infinite and finite-state Markov chains

We can now construct the infinite-state Markov chain $\widehat{M}$ for $\mathcal{S}$.

Definition 4. *$\widehat{M} = (\widehat{S}, \widehat{S}_{in}, \widehat{P})$ where:*

(1) $\widehat{S}_{in} \subseteq TR_\Delta$ is given by: $\mathbf{s} \in \widehat{S}_{in}$ iff $\mathbf{s}(0) \in \mathcal{H}_{in}$

(2) $\widehat{S}$ is the least subset set of TR_Δ satisfying:

- *$\widehat{S}_{in} \subseteq \widehat{S}$*
- *If $\mathbf{s} \in \widehat{S}$ and $\mathbf{s}' \in succ^\Delta(\mathbf{s})$, then $\mathbf{s}' \in \widehat{S}$.*

(3) $\widehat{P}$ is P^Δ restricted to $\widehat{S} \times \widehat{S}$.

Lemma 4. *(1) Suppose $\mathbf{s} \in \widehat{S}$ and $\mathbf{s}(0) = (t, A, R)$. Then there exists $m \geq 0$ such that $t = m\Delta$.*

(2) $\widehat{M}$ is a Markov chain.

Proof. Suppose $\mathbf{s} \in \widehat{S}_{in}$ and $\mathbf{s}(0) = \mathbf{u} = (t, A, R)$. Then $t = 0$ because $\mathbf{u}$ is an initial configuration by the definition of $\widehat{S}_{in}$. Assume inductively that $\mathbf{s} \in \widehat{S}$, $\mathbf{s}(0) = \mathbf{u} = (t, A, R)$ and $t = m\Delta$. If $\widehat{P}(\mathbf{s}, \mathbf{s}') > 0$ and $\mathbf{s}'(0) = \mathbf{v} = (t', A', R')$ then $t' = (m + 1)\Delta$. This follows from the fact that $\mathbf{s}$ is a trace of length Δ and $\mathbf{s}'(0)$ is a successor of $\mathbf{s}(\Delta - 1)$. The first part of the result now follows from the inductive definition of $\widehat{S}$.

Next, we observe that due to the first part of the lemma, $\widehat{S}$ can be partitioned as $\widehat{S} = \{\widehat{S}_0, \widehat{S}_1, \widehat{S}_2 \cdots\}$ where $\widehat{S}_m = \{\mathbf{s} \mid \mathbf{s}(0)(1) = m\Delta\}$. Here $\mathbf{s}(0)(1) = m\Delta$ is the time component of the first configuration in the trace $\mathbf{s}$. According to Prop. 2.1, a configuration can have at most $|C_1| \times |C_2| \times \cdots \times |C_n| = K$ successor configurations. This implies that for every $\mathbf{s} \in \widehat{S}$, $|\{\mathbf{s}' \mid \widehat{P}(\mathbf{s}, \mathbf{s}') > 0\}|$ is of size at most K^Δ. Hence $\widehat{S}_m$ is a finite set for every m, which implies that $\widehat{S}$ is the countable union of finite sets and is hence countable.

From Lemma 3 it now follows that $\widehat{M}$ is a Markov chain. □

We next derive a finite-state irreducible Markov chain representation of the behavior of $\mathcal{S}$. We first define the equivalence relation $\approx$ over the set of configurations and then lift it to $\widehat{S}$. Let $\mathbf{u} = (t, A, R)$ and $\mathbf{v} = (t', A', R')$ be two configurations. Then $\mathbf{u} \approx \mathbf{v}$ iff the following conditions are satisfied:

(1) $(\tau_i, t_i, c_i, d_i) \in A$ iff there exists $(\tau_i, t_i', c_i', d_i') \in A'$ such that $t - t_i = t' - t_i'$, $c_i = c_i'$ and $d_i = d_i'$

(2) $(\tau_i, t_i) \in R$ iff there exists $(\tau_i, t_i') \in R'$ such that $t - t_i = t' - t_i'$

Clearly, $\approx$ is an equivalence relation. Basically, it asserts that the active jobs in two equivalent states have the same release times relative to their current times and in all other respects they are identical. Furthermore, the same holds for all the jobs at rest as well. We let $[\mathbf{u}]$ denote the $\approx$-equivalence class containing $\mathbf{u}$. The following characterization of $\approx$ is key to constructing the finite-state Markov chain M.

Lemma 5. *(1) $\approx$ is of finite index. In other words, $\{[\mathbf{u}] \mid \mathbf{u} \in \mathcal{H}\}$ is a finite set.*

(2) Suppose $\mathbf{u} \approx \mathbf{v}$ and $\mathbf{u} \xrightarrow[1]{p} \mathbf{u}'$. Then, there exists $\mathbf{v}'$ such that $\mathbf{v} \xrightarrow[1]{p} \mathbf{v}'$ and $\mathbf{u}' \approx \mathbf{v}'$.

Proof. Let $\mathbf{u}$ be a configuration. Then we define $\widehat{\mathbf{u}} = (\widehat{A}, \widehat{R})$ where $\widehat{A} = \{(\tau_i, t - t_i, c_i, d_i) \mid (\tau_i, t_i, c_i, d_i) \in A\}$ and $\widehat{R} = \{(\tau_i, t - t_i) \mid (\tau_i, t_i) \in R\}$. We claim that $Z = \{\widehat{\mathbf{u}} \mid \mathbf{u} \in \mathcal{H}\}$ is a finite set. To see this, we first note that there can be only finitely many sets of the form $\widehat{A}$. This is so since the first component of an element of $\widehat{A}$ can take only n different values. Next, the second component of an element can take values only in the set $\{0, 1, \cdots, \max\{T_i\}_i\}$ due to the γ-discretization of the time domain and the definition of a configuration. Next, the third element can take at most $\max\{|C_i|\}_i$ different values. Since $d_i \leq c_i$ for each element $(\tau_i, t_i, c_i, d_i) \in A$, the last element can also only take at most $\max\{|C_i|\}_i$ different values. Similarly, the number of sets of the form $\widehat{R}$ is also finite. Consequently, Z is a finite set. But then from the definition of $\approx$ it follows that $\mathbf{u} \approx \mathbf{v}$ iff $\widehat{\mathbf{u}} = \widehat{\mathbf{v}}$. Thus $\{[\mathbf{u}] \mid \mathbf{u} \in \mathcal{H}\}$ is a finite set.

To show the second part, let $\mathbf{u} \approx \mathbf{v}$ with $\mathbf{u} = (t, A, R)$ and $\mathbf{v} = (t', A', R')$. We first wish to argue that $X_{\mathbf{u}} = X_{\mathbf{v}}$. Suppose $\tau_i \in X_{\mathbf{u}}$. Then there exists $(\tau_i, t_i, c_i, d_i) \in A$ or $(\tau_i, t_i) \in R$ such that $t_i + T_i = t + 1$. This in turn implies that there exists $(\tau_i, t_i', c_i, d_i) \in A'$ or $(\tau_i, t_i') \in R'$ such that $t' - t_i' = T_i - 1$ which in turn implies $\tau_i \in X_{\mathbf{v}}$. Thus $X_{\mathbf{u}} \subseteq X_{\mathbf{v}}$. By a symmetric argument, we can establish $X_{\mathbf{v}} \subseteq X_{\mathbf{u}}$. Hence $X_{\mathbf{u}} = X_{\mathbf{v}}$.

Now suppose $\mathbf{u} \xrightarrow[1]{p} \mathbf{u}'$. Consider the case $X_{\mathbf{u}} = \emptyset$. Then $\mathbf{u} \xrightarrow[1]{1} \mathbf{u}'$ with $\mathbf{u}'$ being the unique successor of $\mathbf{u}$. This follows from Prop. 2.1. Since $X_{\mathbf{v}} = \emptyset$ there exists $\mathbf{v}'$, the unique successor of $\mathbf{v}$ such that $\mathbf{v} \xrightarrow[1]{1} \mathbf{v}'$. It remains to be shown that $\mathbf{u}' \approx \mathbf{v}'$. Let $\mathbf{u}' = (t + 1, B, W)$ and $\mathbf{v}' = (t' + 1, B', W')$. Suppose $(\tau_i, t_i, c_i, d_i) \in B$. Then from the proof of Prop. 2.1, it follows that there exists $(\tau_i, t_i, c_i, d_i) \in A$ with $t_i + T_i > t + 1$. Hence there exists $(\tau_i, t_i', c_i, d_i) \in A'$ such that $t_i' + T_i > t' + 1$. This implies that $(\tau_i, t_i', c_i, d_i) \in B'$ by the definition of $\mathbf{v}'$. Clearly $(t + 1) - t_i = (t' + 1) - t_i'$ since $t - t_i = t' - t_i'$. In a similar fashion we can show that $(\tau_i, t_i', c_i', d_i') \in B'$ implies there exists $(\tau_i, t_i, c_i', d_i') \in B$ such that $(t + 1) - t_i = (t' + 1) - t_i'$.

Now suppose $(\tau_i, t_i) \in W$. Then there exists $(\tau_i, t_i, c_i, 1) \in A$ with $st_i = 1$ or $(\tau_i, t_i) \in R$. In either case, using the fact that $\mathbf{u} \approx \mathbf{v}$, we can conclude that there exists $(\tau_i, t_i') \in W'$ such that $(t + 1) - t_i = (t' + 1) - t_i'$. By a symmetric argument we can also establish that if $(\tau_i, t_i') \in W'$ then there exists $(\tau_i, t_i) \in W$ such that $(t = 1) - t_i = (t' + 1) - t_i'$.

Next assume that $X_{\mathbf{u}} = \{\tau_{i_1}, \tau_{i_2} \cdots, \tau_{i_k}\} = X_{\mathbf{v}}$. Then using Prop. 2.1 and the fact that $X_{\mathbf{u}} = X_{\mathbf{v}}$, it is a laborious but routine exercise to apply the definitions and show that if $\mathbf{u} \xrightarrow[1]{p} \mathbf{u}'$, then there exists $\mathbf{v}'$ such that $\mathbf{v} \xrightarrow[1]{p} \mathbf{v}'$ with $\mathbf{u}' \approx \mathbf{v}'$. By a symmetric argument, we can complete the proof. □

We now lift $\approx$ to $\widehat{S}$. Let $\mathbf{s}, \mathbf{s}' \in \widehat{S}$. Then $\mathbf{s} \equiv \mathbf{s}'$ iff $\mathbf{s}(l) \approx \mathbf{s}'(l)$ for $0 \leq l < \Delta$. As before, we let $[\mathbf{s}]$ denote the $\equiv$-equivalence class containing the state $\mathbf{s}$. It will be convenient to lift Lemma 5 to the setting of states.

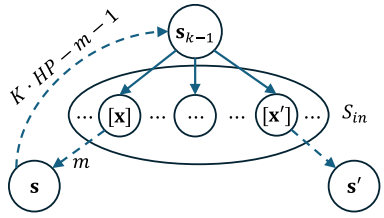

Figure 2: $\mathbf{s}'$ is reachable from $\mathbf{s}$ via $[\mathbf{x}']$.

Lemma 6. *(1) $\equiv$ is of finite index. In other words $\{[\mathbf{s}] \mid \mathbf{s} \in \widehat{S}\}$ is a finite set.*

(2) Suppose $\mathbf{s} \equiv \mathbf{s}'$ and $\mathbf{s} \xrightarrow[\Delta]{p} \mathbf{x}$. Then there exists $\mathbf{x}'$ such that $\mathbf{s}' \xrightarrow[\Delta]{p} \mathbf{x}'$ and $\mathbf{x} \equiv \mathbf{x}'$

Proof. The first part follows easily from the definition of $\equiv$ and the first part of Lemma 5. To show the second part, assume $\mathbf{s}(\Delta - 1) = \mathbf{u}$ and $\mathbf{x} = \mathbf{v}_0\mathbf{v}_1 \cdots \mathbf{v}_{\Delta-1}$. Since $\mathbf{s} \xrightarrow[\Delta]{p} \mathbf{x}$ we have $\mathbf{u} \xrightarrow[1]{p_0} \mathbf{v}_0$ where $p_0 = pr(\mathbf{v}_0)$. Assume $\mathbf{s}'(\Delta - 1) = \mathbf{u}'$. Then $\mathbf{u} \approx \mathbf{u}'$ and hence by lemma 5, there exists $\mathbf{v}_0'$ such that $\mathbf{u}' \xrightarrow[1]{p_0} \mathbf{v}_0'$ which also implies that $pr(\mathbf{v}_0') = p_0$. Furthermore, $\mathbf{v}_0 \approx \mathbf{v}_0'$. Since $\mathbf{x}$ is a trace, $\mathbf{v}_0 \xrightarrow[1]{p_1} \mathbf{v}_1$ where $p_1 = pr(\mathbf{v}_1)$. Since $\mathbf{v}_0 \approx \mathbf{v}_0'$, again by Lemma 5, there exists $\mathbf{v}_1'$ such that $\mathbf{v}_0' \xrightarrow[1]{p_1} \mathbf{v}_1'$ which also implies $pr(\mathbf{v}_1') = p_1$. Furthermore, $\mathbf{v}_1 \approx \mathbf{v}_1'$. Proceeding this way, we can show that for $2 \leq l \leq \Delta - 1$ there exist $\mathbf{v}_l'$ that $\mathbf{v}_l \approx \mathbf{v}_l'$, and $pr(\mathbf{v}_l') = p_l$. Let $\mathbf{x}' = \mathbf{v}_0'\mathbf{v}_1' \cdots \mathbf{v}_{\Delta-1}'$. Then it is easy to see that $\mathbf{s}' \xrightarrow[\Delta]{p} \mathbf{x}'$. Moreover, $\mathbf{x} \equiv \mathbf{x}'$. □

We are now ready to define the required Markov chain.

Definition 5. *$M = (S, S_{in}, P)$ where:*

(1) $S = \{[\mathbf{s}] \mid \mathbf{s} \in \widehat{S}\}$.

(2) $S_{in} = \{[\mathbf{s}] \mid \mathbf{s}(0) \in \mathcal{H}_{in}\}$

(3) $P([\mathbf{s}], [\mathbf{s}']) = p$ if there exists $\mathbf{x} \in [\mathbf{s}]$ and $\mathbf{x}' \in [\mathbf{s}']$ such that $\mathbf{x} \xrightarrow[\Delta]{p} \mathbf{x}'$. Otherwise $P([\mathbf{s}], [\mathbf{s}']) = 0$

Due to Lemma 6, the probabilistic transition function P is well-defined.

Theorem 7. *M is a finite-state irreducible Markov chain.*

Proof. From the first part of Lemma 5 it follows easily that S is a finite set of states.

Next, for $[\mathbf{x}] \in S$, define $succ([\mathbf{x}]) = \{[\mathbf{x}'] \mid P([\mathbf{x}], [\mathbf{x}']) > 0\}$. Now consider $[\mathbf{s}] \in S$. From the definition of $\equiv$ and the second part of Lemma 6, it follows that there exists $\mathbf{x} \in [\mathbf{s}]$ such that $\mathbf{x} \xrightarrow[\Delta]{p} \mathbf{x}'$ iff $\mathbf{x}' \in succ([\mathbf{s}])$. From Lemma 2, it now follows that $\sum_{[\mathbf{s}'] \in succ([\mathbf{s}])} P([\mathbf{s}], [\mathbf{s}']) = 1$. Hence M is a Markov chain.

What remains is to show that M is irreducible. The overall structure of the argument used to establish irreducibility is shown in Fig. 2. The details are as follows.

Let $\mathbf{s} \in S$. Then by the construction of M, there exists $[\mathbf{x}] \in S_{in}$ and a sequence of states $\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_m$ such that $[\mathbf{x}] = \mathbf{s}_0$, $\mathbf{s}_m = \mathbf{s}$, and $P(\mathbf{s}_l, \mathbf{s}_{l+1}) > 0$ for $0 \leq l < m$. In this sense $\mathbf{s}$ is reachable from some $[\mathbf{x}] \in S_{in}$ in M.

Now let $\mathbf{s}' \in S$. By the above reasoning, there exists $[\mathbf{x}'] \in S_{in}$ such that $\mathbf{s}'$ is reachable from $[\mathbf{x}']$ in M. The irreducibility property will follow if we show that $[\mathbf{x}']$ is reachable from $\mathbf{s}$ in M.

Let $HP = \prod_i T_i$ be the hyperperiod of the task system. Let $[\mathbf{z}] \in S_{in}$ and $\rho = \mathbf{z}_0\mathbf{z}_1 \cdots \mathbf{z}_{HP}$ be a path of length HP in $\widehat{M}$ with $\mathbf{z}_0 = \mathbf{z}$. Since $[\mathbf{z}] \in S_{in}$, we must have $\mathbf{z}(0) \in \mathcal{H}_{in}$ and hence will be of the form $(0, A_0, \emptyset)$ with $A_0 = \{(\tau_i, 0, c_i, c_i)\}_i$ where $c_i \in C_i$ for each i. Since each state of $\widehat{M}$ represents a trace of length Δ, we will have, along the path ρ, if $\mathbf{z}_j(0) = (t_j, A_j, R_j)$ then $t_j = j{\cdot}\Delta$. From the definition of HP, it now follows that if $\mathbf{z}_{HP}(0) = (t_{HP}, A_{HP}, R_{HP})$ then $t_{HP} = HP \cdot \Delta$ and hence will be an integer multiple of T_i for each i. This implies that for every task, a new instance of a job belonging to the task will be released at t_{HP}. Furthermore, we have $R_{HP} = \emptyset$ and $A_{HP} = \{(\tau_i, t_{HP}, c_i, c_i\}_i$ with $c_i \in C_i$ for each i. This in turn implies that $[\mathbf{z}_{HP}] \in S_{in}$ by the definitions of S and S_{in}.

Returning to $\mathbf{s}$, let K be the least positive integer such that $m < K \cdot HP$ where m is the length of the path from $[\mathbf{x}]$ to $\mathbf{s}$ in M. Then $k = K \cdot HP - m > 0$. Since M is a Markov chain, each state will have at least one successor state and hence, starting from $\mathbf{s}_0 = \mathbf{s}$ we can construct a path $\mathbf{s}_0\mathbf{s}_1 \cdots \mathbf{s}_{k-1}\mathbf{s}_k$ of length k in M. But then $m + k = K \cdot HP$ and hence by the reasoning above, we will have $\mathbf{s}_k \in S_{in}$. By the definitions of the transition relation of M and S_{in}, each state in S_{in} will be a successor state of $\mathbf{s}_{k-1}$ in M. Hence $\mathbf{s}_0\mathbf{s}_1 \cdots \mathbf{s}_{k-1}[\mathbf{x}']$ will also a path in M. Thus $[\mathbf{x}']$ is reachable from $\mathbf{s}$ in M and the irreducibility of M follows. □

4.3 M represents $B_{\mathcal{S}}$

We wish to argue here that M represents $B_{\mathcal{S}}$. In fact, it will be more convenient to show this for $\widehat{M}$. Due to lemma 6, we can conclude then that this holds for M as well. The basic idea is that there is a $1-1$ correspondence between the set of infinite paths in $\widehat{M}$ that start from an initial configuration and the set of infinite execution sequences. To this end, let $\mathbf{u}_0\mathbf{u}_1 \cdots$ be an infinite execution sequence with $\mathbf{u}_0 \in \mathcal{H}_{in}$ and $\mathbf{u}_l \longrightarrow \mathbf{u}_{l+1}$ for every $l \geq 0$. This implies $\mathbf{u}_l \xrightarrow[1]{p_l} \mathbf{u}_{l+1}$ with $p_l \in (0, 1]$ for every $l \geq 0$. Let $\mathbf{s}_l = \mathbf{u}_l\mathbf{u}_{l+1} \cdots \mathbf{u}_{l+\Delta-1}$ for $l \geq 0$. Then it follows easily that $\mathbf{s}_{l\cdot\Delta} \xrightarrow[\Delta]{p'_l} \mathbf{s}_{(l+1)\cdot\Delta}$ with $p'_l \in (0, 1]$ From the definition of $\widehat{M}$ we can conclude that $\mathbf{s}_0\mathbf{s}_\Delta\mathbf{s}_{2\Delta} \cdots$ is an infinite path in $\widehat{M}$ that starts from the initial state $\mathbf{s}_0$. We let EP be the map that assigns in this way an infinite path σ of $\widehat{M}$ to an infinite execution sequence of ρ of $\mathcal{S}$.

Next let $\sigma = [\mathbf{s}_0][\mathbf{s}_1] \cdots$ be an infinite path in $\widehat{M}$ with $\mathbf{s}_0$ an initial state and $P(\mathbf{s}_l, \mathbf{s}_{l+1}) = p_l > 0$ for every $l \geq 0$. From the definitions of the probabilistic transition relations, it follows that $\mathbf{s}_0\mathbf{s}_1 \cdots$ is an infinite execution sequence of $\mathcal{S}$. We let PE be the map that assigns in this way an infinite execution sequence ρ of $\mathcal{S}$ to an infinite path σ of $\widehat{M}$. It is straightforward to prove that $EP(PE(\sigma)) = \sigma$ for every infinite path σ of $\widehat{M}$ and $PE(EP(\rho)) = \rho$ for every infinite execution sequence ρ of $\mathcal{S}$.

In this sense, $\widehat{M}$ and therefore M represent $B_{\mathcal{S}}$. Consequently, the stationary distribution of M captures long-term average properties of $\mathcal{S}$. Thanks to the ergodic theorem and the equivalence relation $\equiv$, we can start from any initial state of $\widehat{M}$ and sample long enough paths to estimate the desired long-term average. For instance, let $dlm_i : S \longrightarrow \mathbb{N}_0$ be given by $dlm_i(\mathbf{s})$ is the number of deadline misses of jobs belonging to τ_i in the trace represented by $\mathbf{s}$. Then we can estimate the expected value of this quantity through sampling.

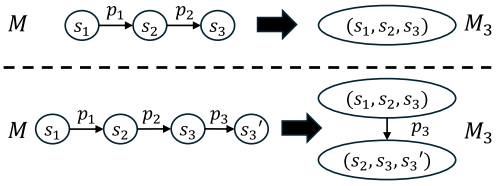

Figure 3: M_3 is induced by M.

4.4 Weakly-hard constraints

Finally, we consider the estimation of the long-term averages of weakly-hard constraints. To be sure, we can sample paths from $\widehat{M}$ and carry forward a bounded amount of information during sampling to estimate averages of weakly-hard constraint violations. However, to establish the statistical validity of such estimates, we must ensure that these averages converge and in the limit they will correspond to the expected numbers of violations defined by the steady state distribution of an irreducible Markov chain. In this case, such sampling-based average estimates can be justified by appealing to the ergodic theorem,

A complication with weakly-hard constraints is that, unlike deadline misses, we can not define a random variable over a *single* state (unit interval) to capture violations since the interval associated with a constraint may straddle two consecutive unit intervals. For instance, with $\Delta = 5$ and $(2, 4)$ as a weakly-hard constraint, the intervals $[2, 6)$, $[3, 7)$ and $[4, 8)$ will straddle the unit intervals $[0, 5)$ and $[5, 10)$. Hence a random variable defined over the states of M (defined in definition.5) will not be able to count the constraint violations occurring during these "straddling" intervals. Consequently, the expected number of violations of the constraint $(2, 4)$ predicted by the stationary distribution of M will not be accurate.

We shall present here our technique for getting around this in a simple setting where the specification consists of a single weakly-hard constraint (m_i, k_i) for each task τ_i. We then fix l to be the least integer such that $\max\{k_i\}_i \leq l$ and define the Markov chain M_l. The states of this chain will be finite paths of length $l - 1$ of M. Further, the transition function of M_l will ensure the following: (i) M_l is an irreducible Markov chain, (ii) we can define, for each i, a random variable wh_i over the states of M_l such that, $\mathbb{E}(wh_i)$, the expected number violations of the constraint (m_i, k_i) defined by the steady state distribution of M_l indeed captures the long term averages of violations of the constraint (m_i, k_i) for each i, and (iii) the estimates of these averages obtained by sampling from $\widehat{M}$ as sketched above, will converge in the limit to $\mathbb{E}(wh_i)$ for each i.

To bring out the main ideas in a sufficiently general setting while minimizing the notational overhead, we shall assume here that $k_i \leq 3 \cdot \Delta$ for each constraint (m_i, k_i). We then define the Markov chain M_3 as follows.

Definition 6. *Let $M = (S, S_{in}, P)$ be as defined in subsection 4.2. Then $M_3 = (S_3, S_3^{in}, P_3)$ where:*

- *$S_3 \subseteq S \times S \times S$ is given by:*
  $(s_1, s_2, s_3) \in S_3$ iff $P(s_j, s_{j+1}) > 0$ for $1 \leq j < 3$.
- *$S_3^{in} \subseteq S_3$ is given by:*
  $(s_1, s_2, s_3) \in S_3^{in}$ iff $s_1 \in S_{in}$.

- *For* $(s_1, s_2, s_3), (s_1', s_2', s_3') \in S_3$:
  $P_3((s_1, s_2, s_3), (s_1', s_2', s_3')) = P(s_3, s_3')$ *if* $(s_2, s_3) = (s_1', s_2')$. *Otherwise,* $P_3((s_1, s_2, s_3), (s_1', s_2', s_3')) = 0$.

The idea underlying this construction is illustrated in Fig. 3. The arguments developed for showing Theorem 7, can be easily extended to show the next result. We omit the details due to space limitations.

Theorem 8. *M_3 is a finite-state irreducible Markov chain.*

We can now define the random variable $wh_i : S_3 \to \mathbb{N}_0$ via: $wh_i((s_1, s_2, s_3)) = k$ iff there are k violations of the constraint (m_i, k_i) in the trace $s_1 s_2 s_3$. Now suppose $\rho = (s_1^0, s_2^0, s_3^0), (s_1^1, s_2^1, s_3^1) \cdots (s_1^j, s_2^j, s_3^j) \cdots$ is an infinite path in M_3. Then (s_1^j, s_2^j, s_3^j) will cover the behavior during $[j{\cdot}\Delta,\ (j{+}3){\cdot}\Delta)$ while $(s_1^{j+1}, s_2^{j+1}, s_3^{j+1})$ will cover the behavior during $[(j{+}1){\cdot}\Delta,\ (j{+}4){\cdot}\Delta)$. This is due to the fact that $(s_2^j, s_3^j) = (s_1^{j+1}, s_2^{j+1})$ by the definition of M_3. Thus wh_i will count *all* the violations of the (m_i, k_i) constraint that occur along ρ and $\mathbb{E}(wh_i)$ defined by the steady state distribution of M_3 will be the long-term average of the number of violations of this constraint. Furthermore, due to the ergodic theorem and the definition of M_3, this boils down to sampling sufficiently long paths from $\widehat{M}$ augmented with some bounded bookkeeping. In this sense, M_3 provides the mathematical basis for estimating the weakly-hard constraint violations through a simple sampling procedure.

The definition of M_3 can be smoothly extended to M_l for any $l \geq 1$. Then given a set of constraints $\{(m_i, k_i)\}_i$, we can choose an l that satisfies $\max(\{k_i\}) \leq l \cdot \Delta$ and define M_l. This will secure the mathematical basis for estimating the long-term averages of the violations of these constraints by sampling paths from $\widehat{M}$. Due to lack of space, we do not present the details here. Finally, our method can be easily extended to handle a rich language of constraints described in [34].

5 Evaluation

To evaluate the performance and practical utility of our method, we conducted a series of experiments using synthetic workloads (Sec. 5.1–5.4) and a real-world rover control system (Sec. 5.5).

Setup. We implemented our Markov chain-based ground truth computation (by explicitly computing the stationary distribution) and the sampling method for computing the long-term average deadline miss and weakly-hard constraint violations. We used the weakly-hard constraint with $(m, k) = (3, 4)$ by default, unless specified otherwise. We implemented both the static-priority policy and the EDF policy (as a representative dynamic scheduling policy). Our implementation was in Python. Experiments were performed on a machine with Intel(R) Xeon(R) Silver 4216 CPUs @ 2.10GHz and 64GB memory. The implementation is available on GitHub (https://github.com/fyc1007261/analysis-LTA).

Workload. We randomly generated periodic task sets of different sizes and utilizations. For each task set with size n and utilization u, we used the Dirichlet-Rescale algorithm [16, 17] to generate n average utilization values $u_1, u_2, \cdots, u_n$ for the n tasks in the set, such that $\sum_{i=1}^{n} u_i = u$. The expectation of the execution time μ_e of a task was set to be the product of its period and average utilization. A task's deadline was set equal to its period. The task set utilization varied between 0.85 and 1.05, at steps of 0.05. For each combination of task set size and utilization, we generated 50 different task sets. In total, we tested $8 \cdot 5 \cdot 50 = 2000$ task sets.

# Tasks	5	6	7	8	9	10
EDF	0.51	4.19	27.12	150.52	2124.34	8222.69
Static	0.50	4.52	24.42	127.17	1888.02	6797.99

Table 1: Median time (seconds) of ground-truth computation.

5.1 Ground truth computation

To establish a reference for evaluating the accuracy of our sampling method, we computed the ground-truth stationary distribution of our Markov chains by using the `scipy` [29] package to compute eigenvectors and eigenvalues. To keep the running times of ground truth computations manageable, the task periods were randomly chosen from $\{3, 4, 6, 12\}$. For each task with expected execution time μ_e, we fixed the distribution of its computation times to be $0.8\mu_e$ with probability 0.5 and $1.2\mu_e$ with probability 0.5.

Results. Table. 1 shows the median time for computing the ground-truth weakly-hard constraint violations. As expected, computing the ground truth is feasible only for small systems; for larger systems, sampling is a much more efficient alternative.

5.2 Sampling methodology

When sampling a path in a Markov chain, we need to know when it is safe to stop. To determine this, we used the diagnostic method proposed by Gelman and Rubin [15] to detect convergence. This method computes a value, referred to as "Rhat ($\hat{R}$) score", which measures convergence by comparing the variance between multiple chains to the variance within each chain. An $\hat{R}$ score close to 1 indicates that the execution sequences sampled are close to convergence, whereas a score much larger than 1 means that more sampling steps are needed. We used the `arviz` library [23] in Python with the recommended `rank` method [32] to compute $\hat{R}$ scores. Besides $\hat{R}$, we also introduced a parameter T_{stab}: only if the $\hat{R}$ scores of all tasks stabilized below a threshold for a sufficiently long time duration T_{stab}, we declared convergence and stopped sampling.

To determine T_{stab} and proper thresholds for $\hat{R}$, we performed experiments on the same task sets used in Sec. 5.1. For all task sets, we sampled four execution sequences (the execution sequences 0–3 in Fig. 4b), measured their violation rates, and computed the $\hat{R}$ score with respect to the number of jobs sampled.

Results. Fig. 4a shows the maximum and mean $\hat{R}$ scores across all tasks in all task sets. We observe that all tasks reach a stable state where $\hat{R} < 1.0002$. Fig. 4b and 4c show the sampling results and the $\hat{R}$ scores during the sampling of a specific task. The violation rates for the execution sequences become stable after around 15000 jobs (Fig. 4b), while the $\hat{R}$ score stabilizes when 10000 jobs are sampled (Fig. 4c). Results for other task sets show a similar relationship, with an approximately 5000 jobs difference, between the convergence of results and $\hat{R}$ score. Therefore, we chose $\hat{R} = 1.0002$ as the threshold and $T_{stab} = 5000$ jobs as the stabilization time.

5.3 Accuracy of the sampling method

We ran the same task sets as in Sec. 5.1 with two distributions of job execution times, using the sampling method discussed above. Besides the distribution described in Sec. 5.1, we also considered a "likely-unlikely" distribution: the job execution time is "likely" to be

Figure 4: (a) Max and mean $\hat{R}$ scores across all tasks in all task sets during sampling. (b) Weakly-hard constraint violation rate of each execution sequence for a specific task. (c) $\hat{R}$ score of the four execution sequences in the middle subfigure.

Job execution time		Uniform distribution									Likely-unlikely distribution								
Difference		Mean value (%)			Median value (%)			99th perc. value (%)			Mean value (%)			Median value (%)			99th perc. value (%)		
Δ Values		4	8	12	4	8	12	4	8	12	4	8	12	4	8	12	4	8	12
Deadline Miss Ratio	EDF	0.031	0.031	0.031	0.000	0.000	0.000	0.411	0.412	0.406	0.043	0.044	0.043	0.023	0.023	0.023	0.290	0.295	0.287
	Static	0.032	0.033	0.032	0.000	0.000	0.000	0.365	0.367	0.351	0.038	0.040	0.038	0.022	0.023	0.021	0.231	0.234	0.236
Weakly-hard Constraints	EDF	0.026	0.027	0.027	0.000	0.000	0.000	0.458	0.439	0.481	0.023	0.023	0.023	0.006	0.006	0.006	0.242	0.243	0.240
	Static	0.039	0.039	0.038	0.000	0.000	0.000	0.518	0.497	0.491	0.025	0.025	0.025	0.004	0.004	0.004	0.233	0.245	0.240

Table 2: Accuracy (by absolute difference) of sampling with job execution time in uniform and likely-unlikely distributions.

$\frac{95}{99}\mu_e$ with probability 0.99, and "unlikely" to be $5\mu_e$ with probability 0.01. For each task set, we computed the absolute difference between the sampling results and the ground truth (Sec. 5.1). We repeated the experiments for different Δ values.

Results. Table 2 shows the sampling accuracy under each of the considered settings. We can make four observations: First, the mean difference is very small, between 0.023% and 0.044% across all evaluated cases, indicating that the sampling method has very high accuracy in general. Further, for the vast majority (99%) of the tasks, the difference between sampling and ground truth results is within 0.518%. Third, under the uniform distribution, the median difference is zero across all settings, which indicates that more than half of the sampling results are the same as the ground truth. Upon a closer examination, we find that many tasks in the system never miss the deadline or violate the weakly-hard constraints. Our sampling method catches this behavior for all of these tasks, hence returning the same estimates as the ground truth. Finally, the results are similar across the different settings of Δ, which suggests that the exact value of Δ has minimal influence on the accuracy. Overall, the results confirm that our sampling method is highly accurate.

5.4 Performance and scalability of sampling

For scalability evaluation, we focused on the four most determinant factors of the Markov chain size: the length of the unit interval Δ, the task set size, the length of task periods, and the number of job execution times. We considered much larger task systems than the ones in earlier experiments. When generating a task set, we first chose an upper bound value for task periods $T_i^{\max}$ (referred to as maximum period value) and then randomly selected integers in $[1, T_i^{\max}]$ as task periods. We performed experiments for a wide range of values for the above four factors, and considered 20 task sets per each combination of values. The results are consistent across these combinations; due to space constraints, we present only the results for variants of the following default values, unless explicitly stated otherwise: $T_i^{\max} = 16$; $\Delta = T_i^{\max}$; task set size is 50; and job execution times are in the range of $[0.8\mu, 1.2\mu]$, with 10 possible execution times which represent a discrete approximation of the Gaussian distribution $\mu = \mu_e$ and $\sigma = 0.2\mu_e$. Specifically, we computed the integral of the probability density function of the Gaussian distribution in each of the 10 slots, and then computed the probability of each of the 10 values based on the integral.

Impact of Δ. We first evaluated the impact of the unit of time Δ on convergence time. We repeated the experiments for different values of Δ, ranging from 1 to 720720 (the hyperperiod), while keeping the default values for other factors. When we tried to draw samples with Δ being the hyperperiod, convergence of some of the task sets was not achieved even after 40 hours; this confirmed a hyperperiod-based analysis is not scalable.

Fig. 5a and Fig. 5b show the convergence time for different Δ settings. We observe that the convergence time is much higher when Δ is either too small or too large, under both scheduling policies. For example, under the static priority policy, the median convergence time is 293.96 s when Δ = 5, 55.65 s when Δ = 500, and 3163.34 s when Δ = 100000. For smaller Δ values, it is often the case that within a Δ-interval, no job is issued or finished, and therefore no useful data is sampled. For large Δ values, even one Δ-interval may be longer than needed for the sampling to converge. Hence our approach allows the user to flexibly choose a value of Δ to improve the sampling performance, instead of fixing the interval at the two extremes of the hyperperiod or the microtick (γ).

Impact of task set size. We next varied the task set size to be in {5, 10, 25, 50, 100, 200, 300, 400, 500, 1000} while keeping the default values for other factors. Fig. 5c and Fig. 5d show the box plots of the convergence time for all task sets grouped by the task set size under EDF and static priority. We observe that the convergence time grows slightly faster than linear but slower than quadratic, and that it is efficient for typical real-time systems. For example, at 50 tasks per set, the median sampling time is less than 45.39 seconds under EDF and 59.82 seconds under static priority policy. At the extreme, for task sets with 1000 tasks, the median time to convergence is 19.53 minutes under EDF, and 32.90 minutes

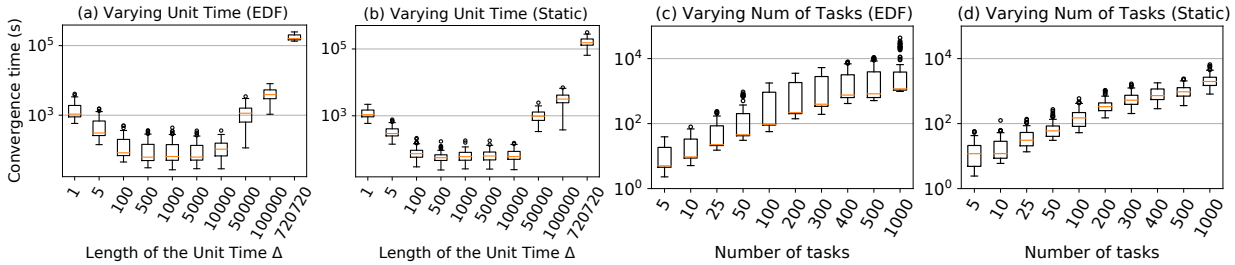

Figure 5: Convergence time of sampling, grouped by unit interval Δ value (a+b) and by the size of the task sets (c+d).

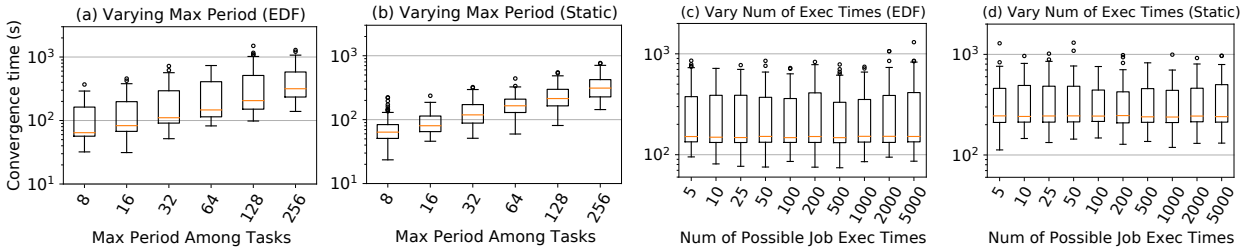

Figure 6: Convergence time, grouped by the maximum length of periods (a+b) and the number of job execution times (c+d).

under the static-priority policy, which is acceptable for offline analysis. In short, our sampling method is efficient for common task set sizes, and it can scale to very large numbers of tasks.

Impact of task periods. We considered a range of *maximum* period values $T_i^{\max}$, chosen randomly in $\{8, 16, 32, 64, 128, 256\}$, and set $\Delta = 128$. For each $T_i^{\max}$, we generated the tasks' periods as integers randomly chosen in the range $[1, T_i^{\max}]$. Fig. 6a and Fig. 6b show the box plots of the convergence time grouped by the longest period for the two scheduling policies. We observe that the convergence time increases as the longest period value increases, but at a slow pace. For instance, increasing the longest period from 8 to 256 (by 32×) leads to around a 5× increase in the mean, median, and maximum analysis time.

Impact of the number of job execution times. This evaluation assesses the efficiency of sampling as we scaled the number of possible job execution times. With the minimum and maximum execution time still being $0.8\mu_e$ and $1.2\mu_e$, respectively, we divided this range into different numbers (instead of the default number 10) of discrete values, which we refer to as *the number of possible job execution times*. We tested the convergence time of our sampling method with the number of possible job execution times in the set of $\{5, 10, 25, 50, 100, 200, 500, 1000, 2000, 5000\}$.

Fig. 6c and Fig. 6d show the convergence time grouped by the number of possible job execution times ($|C_i|$). We observe that as the number of possible job execution times increases, the convergence time does not change much; this means that our analysis scales with the number of values within the job execution time distribution. On a closer examination, we observed that the time taken for picking a job execution time from any distribution during our sampling procedure is insignificant, which explains why the number of possible job execution times has little impact on the analysis time. Thus, by setting a high number of possible job execution times, the discrete job execution time distribution used for sampling can more closely match the actual distribution of execution time, thus improving the accuracy of sampling for real workloads without bringing much overhead to the sampling efficiency.

5.5 Case study: rover control system

To demonstrate the practical utility of our technique, we conducted a case study of a rover control system, `ArduPilot` [2], an open-source autopilot software system for real-time control of unmanned vehicles. `ArduPilot` has been used widely in practice, e.g., by NASA, Intel, Boeing, and in research [1, 4, 5].

Workload and setup. The rover control system contains 45 control tasks (including GPS signal processing, RPM adjustment, etc.) with periods between $2500\mu s$ and $10^7\mu s$. To keep in line with our proposed approach, we modified the scheduler and employed the policy that a job will be killed once it misses the deadline. For the rest, we kept `ArduPilot`'s static-priority and non-preemptive policy. We used SITL (software in the loop) of `ArduPilot` to simulate the environments, sensors, and actuators. The periods (deadlines) and priority values were extracted from `ArduPilot` documentation and source code. All tasks were implemented in C++. We ran the tasks on a single core on a Raspberry Pi 3A+, a common platform for the deployment of ground vehicles.

Execution time profiling. The execution time of a job depends on many factors including the input(s), the current environment, and the recent history of the micro-architectural components. It is difficult to create an accurate model to capture all these factors. Instead, we measured the computation times of the jobs released by a task along a long execution sequence, discretized these times into 10k bins, and extracted a distribution from the resulting histogram. Our working hypothesis was that this distribution is a reasonable approximation from which one can sample the execution times of the jobs belonging to a task.

Accordingly, we ran the entire system with all tasks together on our experimental platform. During the execution, each task released from 120k to 4.8M jobs, depending on its period. The inputs of the tasks varied, reflecting the current state of the real

	Deadline Miss Ratio		(m, k) = (3, 4)		(m, k) = (4, 5)		(m, k) = (8, 10)	
	Observed	Predicted	Observed	Predicted	Observed	Predicted	Observed	Predicted
AHRS Update	0.035%	0.035%	0.006%	0.005%	0.007%	0.006%	0.008%	0.009%
GCS Update Recv	0.055%	0.052%	0.007%	0.006%	0.009%	0.007%	0.009%	0.010%
GCS Update Send	0.057%	0.055%	0.007%	0.006%	0.008%	0.007%	0.009%	0.010%
Gyro FFT Sample	0.005%	0.005%	0.006%	0.005%	0.006%	0.005%	0.009%	0.010%
Gyro FFT Update	0.058%	0.059%	0.008%	0.007%	0.009%	0.009%	0.009%	0.010%
Inertial Sensing	0.058%	0.058%	0.008%	0.007%	0.009%	0.009%	0.009%	0.010%
Set Servos	0.039%	0.040%	0.006%	0.005%	0.007%	0.006%	0.008%	0.009%
Update Mode	0.038%	0.037%	0.006%	0.005%	0.007%	0.006%	0.008%	0.009%
Update Precland	0.057%	0.056%	0.007%	0.007%	0.008%	0.008%	0.009%	0.010%

Table 3: The observed and predicted violation rate comparison in the rover control system.

system. We collected the execution time of each job for our analysis. The measured job execution times of each task formed a (discrete) distribution of its execution time, which we then used as input to our sampling method. Overall, the total system utilizations based on the mean, median, 99.9-th percentile, and 99.99-th percentile execution times of the tasks are 0.15, 0.14, 0.75 and 6.40, respectively.

Predicted results vs. observed results. We applied our sampling method to compute the long-term average weakly-hard constraint violation rates using the profiled execution time distributions as inputs, following the same methodology as in Section 5.2. Further, to evaluate how close our prediction is to the violation rates from actual executions, we executed the rover control system on our experimental platform for 5 runs, 1 hour per run (1.44M jobs for the task with the smallest period), and logged the deadline hits and misses for each task. We then computed the average violation rates for different weakly-hard constraints. Table 3 shows the results computed by our analysis ("predicted") and measured from actual runs ("observed"). (Tasks with both rates below 0.005% are not shown). The results show that the violation rates predicted by our method are very close to the ones observed in real executions. The absolute difference is $\leq 0.003\%$ and $\leq 0.002\%$. This confirms that our analysis can be highly accurate when applied to a real-world system, even when using an approximate execution time distribution.

6 Related Work

There is a rich body of work on probabilistic task systems. The survey [10] provides a broad and detailed account. Here, we focus on research that is closely in line with our work. Accordingly, an often studied property is Deadline Miss Probability (DMP), which is informally stated in [10] as "a probability with a long-run frequency interpretation equating to the expected number of missed deadlines divided by the total number of deadlines in a long (tending to infinite) time interval." This is, however, often formalized in terms of the ratio of the total number of deadline misses to the total number of deadlines in a *hyperperiod*.

The early work on analyzing the DMP of periodic tasks by Woodbury and Shin [36] is also based on hyperperiods termed in their work as major cycles. In addition, most methods for computing DMP and related properties [11, 12, 22, 25, 26] are not only based on hyperperiods but also work with a matrix representation of an associated Markov chain. In contrast, we use a sampling-based approach that avoids the explicit construction of the Markov chain which is computationally infeasible for all but small systems.

A related work aimed at mitigating the effect of the independence assumption while estimating long-term average properties is the multimode Markov model investigated in [14] where each mode is accompanied by a distribution. The analysis however is carried out by first converting the model into an infinite-state Markov chain. Then using an infinite transition matrix, the stationary distribution is estimated using a numerical technique. Consequently, our method can be easily applied to this more general model without appealing to an infinite transition matrix.

Moving a step away from our work, pWCET [13, 31], WCDFP [3, 33] and WCRTEP [8, 27] are properties that have been well studied. However, they are essentially *transient* properties while we focus on long-term averages arising from infinite executions of the system.

Recently, Bozhko et al. [7] proposed a Monte Carlo approach to estimate tasks' response times with much better performance than static analysis (assuming some probability of mis-estimates). This technique is applicable in our setting only when the hyperperiod is chosen as the unit of time so that one can draw independent samples without requiring state-dependent backlogs. However, drawing a sufficient number of samples in this case will be often computationally infeasible.

Weakly-hard constraints have often been studied in co-design techniques, which exploit the ability to tolerate some violations of such constraints to optimize resource usage [19, 24, 35]. Probabilistic analysis for weakly-hard constraints has also been considered [20]. However, we are not aware of any existing work for analyzing long-term average behaviors of probabilistic task systems with weakly-hard constraints.

7 Conclusion and Future Work

We have introduced a Markov chain-based framework for analyzing the long-term average behaviors of probabilistic periodic real-time systems. The novelty of our model is that, unlike previous work, it is not confined to hyperperiods being the unit of time. Our evaluation results have demonstrated that our sampling-based method scales up to large task sets. In our future work, we plan to extend the current theory to settings where the independence assumption about task execution times can be relaxed. In addition, we plan to extend our technique to multiprocessor and distributed systems. It will also be interesting to combine our method with probabilistic formal verification methods [21] to construct a unified framework for analyzing both transient and long-term properties.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thorough and helpful reviews. This work was supported in part by NSF grants CNS-1750158, CNS-1955670, CNS-2111688 and CCF-2326606.

References

[1] Azza Allouch, Omar Cheikhrouhou, Anis Koubâa, Mohamed Khalgui, and Tarek Abbes. 2019. MAVSec: Securing the MAVLink protocol for ardupilot/PX4 unmanned aerial systems. In *Proc. International Wireless Communications & Mobile Computing Conference (IWCMC '19)*.

[2] ArduPilot. 2023. ArduPilot - versatile, trusted, open. https://ardupilot.org/.

[3] Philip Axer and Rolf Ernst. 2013. Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors. In *Proc. Annual Design Automation Conference (DAC '13)*.

[4] Sabur Baidya, Zoheb Shaikh, and Marco Levorato. 2018. FlyNetSim: An open source synchronized UAV network simulator based on ns-3 and ardupilot. In *Proc. International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '18)*.

[5] Simone Baldi, Danping Sun, Xin Xia, Guopeng Zhou, and Di Liu. 2022. ArduPilot-based adaptive autopilot: architecture and software-in-the-loop experiments. *IEEE Trans. Aerospace Electron. Systems* 58, 5 (2022), 4473–4485.

[6] Guillem Bernat, Alan Burns, and Albert Liamosi. 2001. Weakly hard real-time systems. *IEEE transactions on Computers* 50, 4 (2001), 308–321.

[7] Sergey Bozhko, Georg von der Brüggen, and Björn Brandenburg. 2021. Monte carlo response-time analysis. In *Proc. IEEE Real-Time Systems Symposium (RTSS '21)*.

[8] Kuan-Hsun Chen, Mario Günzel, Georg von der Brüggen, and Jian-Jia Chen. 2022. Critical instant for probabilistic timing guarantees: Refuted and revisited. In *Proc. IEEE Real-Time Systems Symposium (RTSS '22)*.

[9] Hoon Sung Chwa, Kang G. Shin, and Jinkyu Lee. 2018. Closing the Gap Between Stability and Schedulability: A New Task Model for Cyber-Physical Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '18)*.

[10] Robert I. Davis and Liliana Cucu-Grosjean. 2019. A Survey of Probabilistic Timing Analysis Techniques for Real-Time Systems. *Leibniz Trans. Embed. Syst.* 6, 1 (2019), 03:1–03:60.

[11] José Luis Díaz, Daniel F García, Kanghee Kim, Chang-Gun Lee, L Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. 2002. Stochastic analysis of periodic real-time systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS '02)*.

[12] Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. 2004. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *Proc. IEEE Real-Time Systems Symposium (RTSS '04)*.

[13] Stewart Edgar and Alan Burns. 2001. Statistical analysis of WCET for scheduling. In *Proc. IEEE Real-Time Systems Symposium (RTSS '01)*.

[14] Bernardo Villalba Frias, Luigi Palopoli, Luca Abeni, and Daniele Fontanelli. 2017. Probabilistic real-time guarantees: There is life beyond the iid assumption. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*.

[15] Andrew Gelman and Donald B Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical science* (1992), 457–472.

[16] David Griffin, Iain Bate, and Robert I. Davis. 2020. dgdguk/drs. https://doi.org/10.5281/zenodo.4118058

[17] David Griffin, Iain Bate, and Robert I Davis. 2020. Generating utilization vectors for the systematic evaluation of schedulability tests. In *Proc. IEEE Real-Time Systems Symposium (RTSS '20)*.

[18] Moncef Hamdaoui and Parameswaran Ramanathan. 1995. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE transactions on Computers* 44, 12 (1995), 1443–1451.

[19] Michael Hertneck, Steffen Linsenmayer, and Frank Allgöwer. 2021. Efficient stability analysis approaches for nonlinear weakly-hard real-time control systems. *Automatica* 133 (2021), 109868.

[20] Eun-Young Kang, Dongrui Mu, and Li Huang. 2018. Probabilistic verification of timing constraints in automotive systems using UPPAAL-SMC. In *Proc. International Conference of Integrated Formal Methods (IFM '18)*.

[21] Joost-Pieter Katoen. 2016. The Probabilistic Model Checking Landscape. In *Proc. ACM/IEEE Symposium on Logic in Computer Science (LICS '16)*.

[22] Kanghee Kim, Jose Luis Diaz, Lucia Lo Bello, José María López, Chang-Gun Lee, and Sang Lyul Min. 2005. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Trans. Comput.* 54, 11 (2005), 1460–1466.

[23] Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo Antonio Martín. 2019. ArviZ a unified library for exploratory analysis of Bayesian models in Python. *Journal of Open Source Software* (2019).

[24] Ching-Chi Lin, Mario Günzel, Junjie Shi, Tristan Taylan Seidl, Kuan-Hsun Chen, and Jian-Jia Chen. 2023. Average task execution time minimization under (m, k) soft error constraint. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '23)*.

[25] José María López, José Luis Díaz, Joaquín Entrialgo, and Daniel García. 2008. Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-Time Systems* 40, 2 (2008), 180–207.

[26] Dorin Maxim, Olivier Buffet, Luca Santinelli, Liliana Cucu-Grosjean, and Robert I Davis. 2011. Optimal Priority Assignment Algorithms for Probabilistic Real-Time Systems. In *Proc. International Conference on Real-Time and Network Systems (RTNS '11)*.

[27] Dorin Maxim and Liliana Cucu-Grosjean. 2013. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *Proc. IEEE Real-Time Systems Symposium (RTSS '13)*.

[28] JR Norris. 1998. *Markov Chains*. Vol. 2. Cambridge University Press.

[29] SciPy. 2024. SciPy: Fundamental algorithms for scientific computing in Python. https://scipy.org/.

[30] Damoon Soudbakhsh, Linh T. X. Phan, Anuradha Annaswamy, Oleg Sokolsky, and Insup Lee. 2013. Co-design of Control and Platform with Dropped Signals. In *Proc. International Conference on Cyber-Physical Systems (ICCPS '13)*.

[31] T-S Tia, Zhong Deng, Mallikarjun Shankar, Matthew Storch, Jun Sun, L-C Wu, and JW-S Liu. 1995. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '95)*.

[32] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2021. Rank-normalization, folding, and localization: An improved R-hat for assessing convergence of MCMC (with discussion). *Bayesian analysis* 16, 2 (2021), 667–718.

[33] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, and Katharina Morik. 2018. Efficiently approximating the probability of deadline misses in real-time systems. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS '18)*.

[34] Nils Vreman, Richard Pates, and Martina Maggio. 2022. WeaklyHard.jl: Scalable Analysis of Weakly-Hard Constraints. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '22)*.

[35] Nils Vreman, Paolo Pazzaglia, Victor Magron, Jie Wang, and Martina Maggio. 2022. Stability of Linear Systems Under Extended Weakly-Hard Constraints. *IEEE Control Systems Letters* 6 (2022), 2900–2905.

[36] Michael H Woodbury and Kang G Shin. 1988. Evaluation of the probability of dynamic failure and processor utilization for real-time systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS '88)*.