Maximizing Utilization under Time-Varying Resource Requirements

YIGE HONG, Carnegie Mellon University, USA
QIAOMIN XIE, University of Wisconsin-Madison, USA
WEINA WANG, Carnegie Mellon University, USA

Low utilization has been one of the key limiting factors for the continued scaling of today's computing systems. One main reason for the low utilization is the temporal variation in the actual resource requirements of jobs, because reserving resources for jobs based on the peak requirements then results in idle resources for non-peak time. To increase utilization, in practice, resources are often overcommitted, so jobs may need to contend for resources at times and experience performance degradation, incurring a cost. To make use of the flexibility in resource allocation provided by overcommitment to the best advantage, it is critical to answer the following fundamental question that has not been well studied: given an acceptable budget for the cost associated with resource contention, how can we achieve the maximum utilization of the system?

In this paper, we propose a job model that captures the time-varying resource requirement of each job by a Markov chain. We consider a stochastic formulation of the job dispatch problem in an infinite-server system. Our goal is to maximize system utilization by minimizing the number of active servers (servers running at least one job), subject to a budget on the cost of resource contention. Our main result is the design of a job dispatch policy that is asymptotically optimal as the job arrival rate increases. The novel technical framework that we develop reduces the policy design problem under study to that in a single-server system through policy conversions, which may be of independent interest. The framework allows us to design the asymptotically optimal policy by solving a linear program.

CCS Concepts: • Mathematics of computing → Queueing theory; Markov processes; • Networks → Network performance analysis.

Additional Key Words and Phrases: System Utilization, Time-Varying Resource, Linear Program

1 INTRODUCTION

In a computing system such as a datacenter, low utilization of the resources on each server means that more servers are required to be running/active to support the same load of demand. Therefore, low utilization results in increased operational costs and energy footprint, and has become one of the key limiting factors for the continued scaling of today’s computing systems [1, 11, 22, 30].

One main cause for low utilization is the temporal variation in the actual resource requirements of jobs. For example, in computing clusters that host user-facing services, a job takes the form of a container or a virtual machine (VM) and it serves requests from users. The actual CPU and memory usage of such a job depends on the instantaneous demand of requests, which varies widely during a day [11]. If the resource manager reserves resources for a job to always satisfy its resource requirement, i.e., reserves resources based on the job’s peak requirement, a significant portion of the reserved resources will idle during non-peak time, leading to low utilization.

In practice, to increase utilization, resources are not reserved based on peak requirements, but rather are overcommitted. Jobs are co-located on a server in a way that the sum of their resource requirements can exceed the server’s resource capacity. When that happens, jobs need to contend for resources, which can result in performance degradation such as violation of Service-Level-Objectives in user-facing applications. However, the idea behind overcommitment is that the cost of resource contention can be justifiable if the utilization is sufficiently improved.

Authors’ addresses: Yige Hong, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 15213; Qiaomin Xie, University of Wisconsin-Madison, Madison, Wisconsin, USA, 53706; Weina Wang, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 15213.
A simplified version of our job model. Each job in service is in either an \( L \) phase or an \( H \) phase, associated with low and high resource requirements, respectively. When the job is completed, it is said to be in the state \( \perp \). The job transitions between the two phases while in service until it is completed, following a continuous-time Markov chain with rates \( \mu_{i'j}, i, i' \in \{L, H, \perp\} \).

(b) A system model with an infinite number of identical servers. As soon as a job arrives to the system, the job needs to be dispatched to a server to start service immediately. The configuration of each server is the number of jobs in each phase on the server.

Fig. 1. Job model and system model.

Now with overcommitment supported by many major platforms [10, 13], it becomes vital to make use of the flexibility in resource allocation provided by overcommitment to the best advantage. In particular, it is critical to answer the following fundamental question:

*Given an acceptable budget for the cost associated with resource contention, how can we achieve the maximum utilization of the system?*

Existing theoretical work on maximizing utilization has not tackled time-varying job resource requirements, but rather has focused on reducing resource fragmentation [9, 14, 24, 25, 28, 29, 32, 34–36]. There, the goal is to pack as many jobs on each server as possible so the total number of active servers is minimized, assuming that each job has a fixed resource requirement and each server’s resource capacity cannot be violated. In the line of work [32, 34–36], the authors consider a stochastic formulation and develop various algorithms (with different complexities) to asymptotically minimize objectives related to the number of active servers. Buchbinder et al. [9] consider a worse-case formulation and develop algorithms with approximation guarantees, both in an offline setting and in an online setting with additional demand information provided by machine learning tools. Another line of work [24, 25, 28, 29] studies the utilization problem from a throughput perspective, where the system has a fixed finite number of servers and jobs are allowed to queue when there are not enough resources. The authors develop scheduling algorithms with throughput performance guarantees. Our work differs from existing work by considering the first stochastic model that captures time-varying resource requirements.

**Problem Formulation**

We first describe our job model that features time-varying resource requirements. For ease of exposition, here we assume that each job in service can be in one of the two phases, \( L \) and \( H \), associated with a low resource requirement and a high resource requirement, respectively. Our full model allows more than two phases and is presented in Section 3. To model the temporal variation in the resource requirement, we assume that each job transitions between the two phases while in service until it is completed, following a continuous-time Markov chain illustrated in Figure 1(a). We use an absorbing state \( \perp \) to denote that the job is completed. A job can initialize in either phase \( L \) or phase \( H \), and they are referred to as type \( L \) and type \( H \) jobs, respectively.
We then consider a system with an infinite number of identical servers, illustrated in Figure 1(b). We assume that the two types of jobs arrive to the system following two independent Poisson processes, with rates $\Lambda_L$ and $\Lambda_H$, respectively. As soon as a job arrives, it needs to be dispatched to a server according to a dispatch policy, and the job enters service immediately.

Under this model, system utilization and cost of resource contention of each dispatch policy $\sigma$ are measured as follows, assuming that the policy induces a stable system with a stationary distribution. System utilization is measured by the expected number of active servers (servers currently serving a positive number of jobs) in steady state, denoted as $N(\sigma)$. To quantify the cost of resource contention on a server, we first represent the state of a server by its configuration, a vector $k = (k_L, k_H)$ where $k_L$ and $k_H$ are the numbers of jobs in phase $L$ and phase $H$, respectively. Then a cost rate function $h(\cdot)$ maps a server’s configuration to a rate of cost. Typically, the cost rate is positive when the total resource requirement of all jobs on a server exceeds the server’s resource capacity, although our assumptions allow more general forms of $h(\cdot)$. Here we assume that the resource contention does not affect the transition rates in the job model nor prompt jobs to be killed, suitable for the application scenarios where contention level is low and manageable. Let $C(\sigma)$ denote the average expected cost rate per server.

Now the utilization maximization problem can be formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad N(\sigma) \\
\text{subject to} & \quad C(\sigma) \leq \epsilon,
\end{align*}
\]

where $\epsilon$ is a budget for the cost of resource contention. We are interested in solving this problem in the asymptotic regime where the arrival rates $(\Lambda_L, \Lambda_H)$ scale to infinity [32, 34–36], motivated by the ever-increasing computing demand that drives today’s computing systems to be large-scale. Specifically, we assume $(\Lambda_L, \Lambda_H) = (\lambda_L r, \lambda_H r)$ for some fixed coefficients $\lambda_L$ and $\lambda_H$ and a scaling factor $r \to \infty$.

**Main Result**

We propose a meta-policy named JOIN-THE-RECENTLY-REQUESTED-SERVER (JRRS) for dispatching jobs in the considered system. JRRS can take different subroutines as input, and we develop a specific subroutine named SINGLE-OPT. We give a brief description of JRRS and SINGLE-OPT in the Technical Approach below, and leave the detailed description to Section 6 and Section 7. We prove that JRRS with the SINGLE-OPT subroutine is asymptotically optimal, in the sense that the expected number of active servers is at most $(1 + O(r^{-0.5}))$ times the optimal value of the optimization problem in (1), while the cost rate incurred is at most $(1 + O(r^{-0.5})) \cdot \epsilon$. This main result is presented in Theorem 1.
Technical Approach

We develop a novel policy-conversion framework that reduces the optimal-policy design problem in (1) to an optimal-policy design problem in a single-server system. Specifically, this single-server system, as illustrated in Figure 2, has an infinite supply of jobs of all types. As a result, the server can start the service of any number of new jobs of any type at any time. When the server starts the service of a new job, we say that the server requests the job (from the infinite supply). We assume that the server is also subject to a resource contention cost, following the same cost model as that in the original infinite-server system. Then the optimal-policy design problem in this single-server system is as follows. We aim to find a policy that decides when to request jobs of each type, such that the system throughput is equal to \( \left( \frac{\Lambda_L}{N}, \frac{\Lambda_H}{N} \right) \) for some number \( N \) as small as possible, while the cost rate is kept within a budget \( \epsilon \). The objective \( N \) can be interpreted as the copies of the single-server system needed so that when they run in parallel, their total throughput matches the arrival rates \( (\Lambda_L, \Lambda_H) \) in the infinite-server system. For ease of notation, we simply write \( N \) to mean \( \lceil N \rceil \), the smallest integer number larger than or equal to \( N \). Let \( N^* \) be the optimal value for \( N \). Our approach has the following three steps.

**Lower Bound.** We first show that the expected number of active servers in the original infinite-server system is lower bounded by \( N^* \). This result is formally presented in Theorem 2. The proof is carried out using a coupling argument. Intuitively, the single-server system can always be made more efficient than an “average” server in the infinite-server system, since the former can mimic the dynamics of any server in the latter system.

**Converting a Single-Server Policy to an Infinite-Server Policy.** Next, we come to the key part of our framework, where we use the proposed meta-policy JRRS to convert a policy \( \overline{\sigma} \) in the single-server system to a dispatch policy in the original infinite-server system. We say that JRRS takes \( \overline{\sigma} \) as a subroutine. Under this conversion, we prove that the \( \overline{N} \) under \( \overline{\sigma} \) corresponds to the expected number of active servers under the converted policy in the infinite-server system, and that the converted policy conforms to the budget \( \epsilon \), all in an asymptotic sense. This result is presented in Theorem 3, which we refer to as the conversion theorem. With the conversion theorem, designing an asymptotically optimal policy in the original infinite-server system reduces to designing a policy in the single-server system that achieves the optimal value \( \overline{N^*} \).

We outline this conversion procedure of JRRS below. Fix a policy \( \overline{\sigma} \) for the single-server system. We let each server \( \ell \) with \( \ell \in \{1, 2, \ldots, \overline{N} \} \) in the infinite-server system adopt \( \overline{\sigma} \), based on the current configuration of server \( \ell \). However, when \( \overline{\sigma} \) requests a job in the single-server policy, server \( \ell \) cannot obtain a job immediately in the infinite-server system. In this case, server \( \ell \) generates a token and sends it to the dispatcher to store the request. The type of the token is set to be the type of the requested job. Then when a job of type \( i \) (e.g., \( i \in \{L, H\} \)) arrives to the system, the dispatcher dispatches the job based on the tokens of type \( i \). In particular, the dispatcher chooses a type \( i \) token uniformly at random, sends the job to the server that generates the token, and removes the token; when there are no type \( i \) tokens, the dispatcher sends the job to a server with an index greater than \( \overline{N} \).

Under this conversion, although each server \( \ell \in \{1, 2, \ldots, \overline{N} \} \) in the infinite-server system cannot obtain a requested job immediately as in the single-server system, it should be able to receive the job with a diminishing delay since the arrival rates of jobs scale up to infinity. Therefore, the dynamics of server \( \ell \) can be approximated by that of a single-server system. We formally prove this approximation through Stein’s method [6–8].
**Optimal Single-Server Policy.** Now the last piece of the proposed conversion framework is to develop a single-server policy $\pi$ that achieves the optimal value $N^*$. This task is typically much simpler than finding an optimal policy in the infinite-server system. We design a single-server policy, SINGLE-OPT, through a linear program based on state-action frequencies. The optimality is proved in Theorem 4.

Combining the three steps above, we establish the asymptotic optimality of the meta-policy JRRS with the SINGLE-OPT subroutine. We comment that as stated in the conversion theorem, our policy-conversion framework allows us to convert not only SINGLE-OPT but rather a general class of single-server policies to infinite-server policies with performance guarantees. This framework also avoids the typically tedious work involved in fluid-limit based approaches such as justifying the validity of fluid limit for each policy.

2 RELATED WORK

Most existing theoretical work on maximizing utilization has focused on reducing resource fragmentation [9, 14, 24, 25, 28, 29, 32, 34–36], and has not tackled time-varying job resource requirements. The line of work closest to our work is [14, 32, 34–36]. This line of work also looks into the job dispatch problem in systems with an infinite number of servers, and aims to minimize the total number of active (occupied) servers. Each paper in this line of work proves a certain notion of asymptotic optimality for the policy under consideration. In [34], the authors propose a policy that asymptotically minimizes the number of active servers when arrival rates scale to infinity. The disadvantage of the policy in [34] is that it has to query the configuration of all servers before dispatching a job. To address this issue, the work in [35, 36] studies a simpler policy, which is shown to be asymptotically optimal in [36]. The work [32] extends the above model to the setting with heterogeneous packing constraints. A more practical policy based on the commonly used Best-Fit bin packing heuristics is studied in [14]. We reiterate that jobs in these papers have fixed resource requirements throughout their lifetimes. The analysis in this line of work is typically per policy based: for each proposed policy, the authors establish its fluid limit and prove that the equilibrium of the fluid limit captures the asymptotic behavior of the policy. In contrast, our proof framework allows us to obtain the performance characterization of a large class of policies directly.

In other work that focuses on reducing resource fragmentation, [9] considers a worse-case formulation and develops algorithms with approximation guarantees. Another line of work [24, 25, 28, 29] studies the utilization problem by maximizing throughput in a system with a finite number of servers and buffer space for waiting jobs.

The notion of time-varying job resource requirement has been considered in a variant of Resource-Constrained Project Scheduling Problem (RCPSP) [18–20, 38]. However, apart from the common feature of time-varying resource requirement, the settings studied in this line of work are very different from ours. In particular, they consider a finite set of jobs with known duration, and their objective function is makespan instead of utilization.

The class of JRRS policies proposed in this paper uses tokens, and thus shares some common features with token-based job dispatch policies in the load balancing literature [12, 15, 16, 23, 27, 31, 33]. Specifically, the tokens in these policies allow the servers to inform the dispatcher that they are ready to accept new jobs, instead of relying on the dispatcher to query the servers. As a result, the communication delay is low, which is desirable for large systems.

Another related but different model is the parallel job model with a server pool. In particular, jobs require a fixed number of servers [17, 21, 37], or experience a speedup proportional to the number of servers they run on [2–5]. The goal there is not to maximize utilization, but to optimize throughput or latency.
3 MODEL

Job Model. As described in the Introduction, we consider a job model where each job in service can be in one of multiple phases, each phase associated with a different resource requirement. To model the temporal variation in the resource requirement, we assume that each job transitions between phases while in service until it is completed. The phase transition process is described by a continuous-time Markov chain on the state space $I \cup \{\bot\}$, where $I$ is the set of phases and $\bot$ is the absorbing state that denotes the completion of the job. We assume that any two phases in the Markov chain communicate with each other. We call a transition between two phases in $I$ an internal transition, and let $\mu_{i,i'}$ denote the transition rate from phase $i$ to phase $i'$; the departure of a job then corresponds to a transition from a phase $i \in I$ to $\bot$, whose transition rate is denoted as $\mu_{\bot}$.

The phase transitions of different jobs are assumed to be independent of each other.

We classify a job as a type $i$ job if it starts from phase $i \in I$ when entering service. Jobs of each type $i$ arrive to the system according to an independent Poisson process with rate $\Lambda_i$. Suppose that the system is in state $(k_t)_{t \in \mathbb{Z}_+}$. Let $e_i$ be an $|I|$-dimensional vector whose $i$-th entry is 1 and all other entries are 0. Then the following state transitions can happen:

- $k^t \rightarrow k^t + e_i$, and $k^{t'} \rightarrow k^{t'}$ for all $t' \neq t$: a job of type $i$ arrives and is dispatched to server $t$;
- $k^t \rightarrow k^t + e_{i'} - e_i$, and $k^{t'} \rightarrow k^{t'}$ for all $t' \neq t$: a job on server $t$ transits from phase $i$ to phase $i'$;
- $k^t \rightarrow k^t - e_i$, and $k^{t'} \rightarrow k^{t'}$ for all $t' \neq t$: a job on server $t$ departs the system from phase $i$.

The specifics of the system dynamics depend on the employed dispatch policy that decides which server to dispatch to when a job arrives.

Utilization and Active Servers. We measure system utilization by the number of active servers, i.e., servers currently serving a positive number of jobs. Given the arrival rates of jobs, the smaller the number of active servers, the better the system is utilized. Let $X_k(t)$ be the number of servers in configuration $k$ at time $t$, i.e.,

$$X_k(t) = \sum_{t=1}^{\infty} \mathbb{I}_{\{K'(t)=k\}}.$$

Then the number of active servers can be written as $\sum_{k \neq 0} X_k(t)$, where $0 \in \mathbb{R}^{|I|}$ is the zero vector.

Cost of Resource Contention. When a server overcommits its resources, the server may admit many (but no more than $K_{\text{max}}$) jobs into service, and the total resource requirement of the jobs on the server can exceed the server’s resource capacity $M$. Although overcommitment helps improve utilization, it incurs a certain cost when jobs on the same server need to contend for resources and thus experience performance degradation. To quantitatively understand the relationship between
utilization and resource contention, we formalize the cost of resource contention below. Suppose that a job in phase \(i\) has a resource requirement of \(m_i\). Then the total resource requirement on a server in configuration \(k\) is \(\sum_{i \in \mathcal{I}} m_i k_i\). When \(\sum_{i \in \mathcal{I}} m_i k_i > M\), we assume that the server experiences a cost at rate \(h(k)\), where \(h: \mathcal{K} \to \mathbb{R}\) is called the cost rate function. For example, \(h(\cdot)\) can be a piecewise linear function \(h(k) = (\sum_{i \in \mathcal{I}} m_i k_i - M)^+\). More generally, we assume that \(h(\cdot)\) is any function that is \(\Gamma\)-Lipschitz continuous with respect to the \(L^1\) distance for some constant \(\Gamma > 0\) and satisfies \(h(0) = 0\).

**Performance Goal.** Our high-level goal is to design dispatch policies that minimize the number of active users while keeping the cost rate of resource contention within a certain budget. Specifically, we consider policies that are allowed to be randomized and non-Markovian (i.e., the policies can make history-dependent decisions). We further focus on policies that induce a unique stationary distribution on the configuration process \(\{(\mathcal{K}_t^i(t))_{t \in \mathbb{Z}_+}\}\) assuming that the configuration process is embedded in a Markov chain that has a stationary distribution. We are interested in such policies because the resulting time averages of quantities related to the configurations are equal to the corresponding expectations under the unique stationary distribution regardless of the initial state. Let \(\sigma\) be a policy of interest, \((\mathcal{K}_t^i(t))_{t \in \mathbb{Z}_+}\) be a random element that follows the stationary distribution of the system state induced by \(\sigma\), and \(X_k\) be the corresponding number of servers in configuration \(k\) in steady state under \(\sigma\). Then the expected number of active servers is given by

\[
N(\sigma) \triangleq \sum_{k \neq 0} \mathbb{E}[X_k],
\]

and the average expected cost rate per active server is given by

\[
C(\sigma) \triangleq \frac{\sum_{k \neq 0} h(k) \mathbb{E}[X_k]}{\sum_{k \neq 0} \mathbb{E}[X_k]}.
\]

Now our goal can be formulated as the following optimization problem, referred to as \(\mathcal{P}((\Lambda_i)_{i \in \mathcal{I}}, \epsilon)\):

\[
\begin{align*}
\text{minimize} & \quad N(\sigma) \\
\text{subject to} & \quad C(\sigma) \leq \epsilon,
\end{align*}
\]

where \(\epsilon\) is a budget for the cost of resource contention.

**Asymptotic Optimality.** We focus on the asymptotic regime where for all \(i \in \mathcal{I}\), the arrival rate \(\Lambda_i = \lambda_i r\) for some constant coefficient \(\lambda_i\) and a positive scaling factor \(r \to +\infty\). To define asymptotic optimality, we first define the following notion of approximation to the optimization problem \(\mathcal{P}((\Lambda_i)_{i \in \mathcal{I}}, \epsilon)\) in (2): a policy \(\sigma\) is said to be \((\alpha, \beta)\)-optimal if \(N(\sigma) \leq \alpha \cdot N^*(((\Lambda_i)_{i \in \mathcal{I}}, \epsilon))\) and \(C(\sigma) \leq \beta \cdot \epsilon\), where \(N^*(((\Lambda_i)_{i \in \mathcal{I}}, \epsilon))\) is the optimal objective value in (2). Now consider a family of policies \(\sigma^{(r)}\) indexed by the scaling factor \(r\). We say that the policy \(\sigma^{(r)}\) is **asymptotically optimal** if it is \((\alpha^{(r)}, \beta^{(r)})\)-optimal to the optimization problem \(\mathcal{P}((\lambda_i r)_{i \in \mathcal{I}}, \epsilon)\) with \(\alpha^{(r)}, \beta^{(r)} \to 1\) as \(r \to \infty\). We will suppress the superscript \((r)\) for simplicity when it is clear from the context.

## 4 OVERVIEW OF MAIN RESULT AND OUR APPROACH

### 4.1 Main Result

Our main result is the asymptotic optimality of our proposed policy *Join-the-Recently-Requested-Server* (JRRS), with a subroutine we call *Single-OPT*, as briefly discussed in Section 1. We give the detailed descriptions of JRRS and *Single-OPT* to Section 6 and Section 7.

**Theorem 1 (Asymptotic Optimality).** Consider an infinite-server system that serves jobs with time-varying resource requirements. Let the arrival rates be \((\lambda_i r)_{i \in \mathcal{I}}\) and the cost rate budget be \(\epsilon\).
Then the policy \textit{Join-the-Recently-Requested-Server (JRRS)} with the subroutine \textit{Single-OPT} is \((1 + O\left(r^{-0.5}\right), 1 + O\left(r^{-0.5}\right))\)-optimal. That is, the expected number of active servers under JRRS with Single-OPT is at most \((1 + O\left(r^{-0.5}\right))\) times the optimal value of the problem \(\mathcal{P}(\lambda_i r)_{i \in I}, \epsilon)\), while the cost rate incurred is at most \((1 + O\left(r^{-0.5}\right)) \cdot \epsilon\).

The proof of Theorem 1 is given at the end of Section 4.2. Before that, we first introduce our approach and state the results needed for the proof.

### 4.2 Our Approach

In a nutshell, our approach is to reduce the original optimization problem in an infinite-server system to an optimization problem in a single-server system, which is defined below.

#### A Single-Server System.

Consider a single-server system serving jobs with time-varying resource requirements. The system has an infinite supply of jobs of all types. As a result, the server can request any number of new jobs of any type at any time. Once a job is requested, it immediately enters service.

We represent the server configuration at time \(t\) using a vector \(\mathbf{K}(t) = (K_i(t))_{i \in I}\), whose \(i\)-th entry denotes the number of jobs in phase \(i\). We assume that the single-server system has the same service limit \(K_{\text{max}}\) and cost rate function \(h(\cdot)\) as a server in the original infinite-server system. Therefore, the server configuration \(\mathbf{K}(t)\) is also in the set \(K = \{k: \sum_{i \in I} k_i \leq K_{\text{max}}\}\), and the cost rate at time \(t\) is \(h(\mathbf{K}(t))\).

A single-server policy \(\sigma\) determines when to request jobs and how many jobs of each type to request. We allow the policy to be randomized and make decisions based on the current configuration and the history. Let \(\pi(k))_{k \in K}\) be a stationary distribution of the server configuration under the policy \(\sigma\), and let \(\mathbf{K}(\infty)\) be a random variable following the stationary distribution. When we consider a policy \(\sigma\) and its stationary distribution \(\pi\), we assume that the system is initialized from \(\pi\). The policy \(\sigma\) together with \(\pi\) defines the request rate of type \(i\) jobs \(\lambda_i\), which is the expected number of type \(i\) jobs requested per unit time in steady state. Note that \(\lambda_i\) is the throughput of type \(i\) jobs since the system has a finite state space.

We consider the following single-server optimization problem, denoted as \(\overline{\mathcal{P}}((\lambda_i r)_{i \in I}, \epsilon)\):

\[
\begin{align*}
\text{minimize} & \quad \overline{N} \\
\text{subject to} & \quad \mathbb{E}\left[h(\mathbf{K}(\infty)) | \mathbf{K}(\infty) \neq 0\right] \leq \epsilon, \quad \overline{N} \cdot \overline{\lambda}_i = \lambda_i r, \quad \forall i \in I.
\end{align*}
\]

The single-server optimization problem can be interpreted as follows. We can think of \(\overline{N}\) as the number of copies of the single-server system under \(\overline{\sigma}\) needed to support the arrival rates \((\lambda_i r)_{i \in I}\) in the infinite-server system. To minimize \(\overline{N}\), it is equivalent to maximizing the throughput \((\overline{\lambda}_i)_{i \in I}\) in each single-server system, while maintaining their proportions.

We remark that for the problem \(\overline{\mathcal{P}}((\lambda_i r)_{i \in I}, \epsilon)\), we only need to consider policies that do not depend on the scaling factor \(r\). To see this, we can replace the decision variable \(\overline{N}\) with \(\bar{n} \triangleq \overline{N}/r\) and the optimization problem can be equivalently formulated as follows, which does not involve \(r\):

\[
\begin{align*}
\text{minimize} & \quad \bar{n} \\
\text{subject to} & \quad \mathbb{E}\left[h(\mathbf{K}(\infty)) | \mathbf{K}(\infty) \neq 0\right] \leq \epsilon, \quad \bar{n} \cdot \overline{\lambda}_i = \lambda_i, \quad \forall i \in I.
\end{align*}
\]
**Lower Bound.** The single-server optimization problem provides us with a lower bound to the original problem given in (2) as stated in the following theorem. The proof is given in Section 5.

**Theorem 2 (Lower Bound).** Consider an infinite-server system that serves jobs with time-varying resource requirements. Let the arrival rates be \((\lambda_i r_i)_{i \in I}\) and the cost rate budget be \(e\). Let \(N^*\) be the optimal value of the original infinite-server problem in (2), and let \(\overline{N^*}\) be the optimal value of the single-server optimization problem \(\overline{P}(\lambda_i r_i)_{i \in I}, e\), then

\[
N^* \geq \overline{N^*}.
\]  

**Converting From the Single-Server System to the Infinite-Server System.** Having established a lower bound on the infinite-server problem \(P(\lambda_i r_i)_{i \in I}, e\) in terms of the optimal value of the single-server problem \(\overline{P}(\lambda_i r_i)_{i \in I}, e\), next we focus on finding an asymptotically optimal policy. In particular, we will characterize the performance guarantee of a class of policies and then show that the best policy within the class is asymptotically optimal. Specifically, we consider a meta-policy called JOIN-THE-RECENTLY-REQUESTED-SERVER (JRRS), which converts a Markovian single-server policy \(\sigma\) into an infinite-server policy. We call the policy resulting from the conversion a JRRS policy with a subroutine \(\overline{\sigma}\). Through analyzing the meta-policy JRRS, we show that the performance of each JRRS policy can be characterized by the performance of its subroutine, as stated below.

**Theorem 3 (Conversion Theorem).** Consider an infinite-server system that serves jobs with time-varying resource requirements. Let the arrival rates be \((\lambda_i r_i)_{i \in I}\) and the cost rate budget be \(e\). Let \((N, \overline{\sigma}, \pi(k))\) be a solution feasible to the single-server optimization problem \(\overline{P}(\lambda_i r_i)_{i \in I}, e\). In addition, we assume that the policy \(\overline{\sigma}\) is Markovian. Let the infinite-server policy \(\sigma\) be JRRS with a subroutine \(\overline{\sigma}\). Then under \(\sigma\), we have

\[
\sum_{k \neq 0} E \left[ X_k^{(r)} \right] - \overline{N} \cdot P (K \neq 0) = O (r^{0.5}), \tag{6}
\]

\[
\sum_{k \neq 0} h(k) E \left[ X_k^{(r)} \right] - \overline{N} \cdot E \left[ h(K) \right] = O (r^{0.5}). \tag{7}
\]

As a result,

\[
N(\sigma) \leq (1 + O (r^{-0.5})) \cdot \overline{N}, \tag{8}
\]

\[
C(\sigma) \leq (1 + O (r^{-0.5})) \cdot e. \tag{9}
\]

We briefly describe JRRS here. Let each server with index \(\ell \leq \overline{N}\) request new jobs based on its configuration, according to the input single-server policy \(\overline{\sigma}\). A request for a type \(i\) job is sent to the centralized dispatcher in the form of a type \(i\) token. When a type \(i\) job arrives, the dispatcher randomly chooses a type \(i\) token, removes the token, and sends the job to the server that generates the token; when there are no type \(i\) tokens, the dispatcher sends the job to a server with index \(\ell > \overline{N}\). The full description of JRRS is presented in Section 6, which includes additional details that we omit here.

Under JRRS with a subroutine \(\overline{\sigma}\), the configurations of the first \(\overline{N}\) servers \((K^1(t), \ldots, K^{\overline{N}}(t))\) have approximately the same steady-state distribution as \(\overline{N}\) independent and identical copies of the single-server system under \(\overline{\sigma}\). Moreover, the number of active servers with index greater than \(\overline{N}\) is diminishing with \(r\). These two facts allow us to connect the performance metrics of the infinite-server problem (2) and the single-server problem (3). The intuition behind these facts is that when the arrival rate goes to infinity, each server should be able to receive the requested jobs with diminishing delays, making them behave like independent copies of the single-server system.
Optimal Single-Server Policy. Theorem 3 together with the lower bound in Theorem 2 reduces the infinite-server optimization problem $\mathcal{P}((\lambda_i)_{i \in I}, \epsilon)$ in (2) to the single-server optimization problem $\mathcal{P}((\lambda_i r_{i})_{i \in I}, \epsilon)$ in (3). To solve the single-server optimization problem, we consider the following linear program denoted as $\mathcal{LP}((\lambda_i)_{i \in I}, \epsilon)$ with decision variables $\Phi \in \mathbb{R}$, $\pi \in \mathbb{R}^{|K|}$, and $u_i \in \mathbb{R}^{|K|}$ for $i \in I$. The linear program (LP) is derived as a relaxation to the optimization problem (3), and $\pi \in \mathbb{R}^{|K|}$ and $u_i \in \mathbb{R}^{|K|}$ in the solution will be used to construct a single-server policy.

\[
\begin{align*}
\text{maximize} & \quad \Phi, \pi, (u_i)_{i \in I} \\
\text{subject to} & \quad h^T \pi \leq \epsilon (1 - \pi_0) \\
& \quad 1_k^T u_i = \Phi \cdot \lambda_i \quad \forall i \\
& \quad A\pi + \sum_{i \in I} B_i u_i = 0 \\
& \quad 1^T \pi = 1 \\
& \quad \pi \geq 0, u_i \geq 0 \quad \forall i \in I
\end{align*}
\]

where $h$ is the vector form of the cost rate function $h$; $1_k^T$ is a $|K|$-dimensional vector with one in all entries except those with $\sum_{i \in I} k_i = K_{\text{max}}$. The linear equation $A\pi + \sum_{i \in I} B_i u_i = 0$ is explicitly given by

\[
\begin{align*}
\sum_i u_i(k - e_i)1_{\{k_i \geq 1\}} + & \sum_i (k_i + 1)\mu_{i,k} \pi(k + e_i) + \sum_{i,i',i \neq i'} (k_i + 1)\mu_{i,k} \pi(k + e_i - e_{i'})1_{\{k_{i'} \geq 1\}} \\
= & \sum_i u_i(k) + \left( \sum_i k_i\mu_{i,k} + \sum_{i,i',i \neq i'} k_i\mu_{i,k} \right) \pi(k), \quad \forall k \in K.
\end{align*}
\]

The decision variables and constraints of the problem have the following interpretations: $\Phi$ is a factor that scales inversely with the request rates of the system; $\pi$ is a column vector representing the stationary distribution of configuration; $u_i$ will be referred to as transition frequency, whose $k$-th entry $u_i(k)$ represents the number of transitions from configuration $k$ to configuration $k + e_i$ per unit time in steady state, indicating how often type $i$ jobs should be requested when the system is in configuration $k$. The first two constraints of the linear program correspond to the first two constraints in $\mathcal{P}((\lambda i r_{i})_{i \in I}, \epsilon)$, and the last three constraints characterize the fact that $\pi$ is a stationary distribution.

Given a solution $(\Phi, \pi, (u_i)_{i \in I})$ feasible to the linear program in (10), we construct a single-server policy using Poisson clocks. In particular, if the system enters a configuration $k$ with $\pi(k) \neq 0$, the policy runs a Poisson clock with rate $u_i(k) / \pi(k)$ for each $i \in I$, and requests a type $i$ job when the corresponding clock ticks. Note that the clock rate gives us the desired transition frequency in steady-state because $u_i(k) / \pi(k)$ is the clock rate. The case where $\pi(k) = 0$ is dealt with separately. The detailed description of the algorithm is given in Section 7. We call such single-server policies LP-based policies and the LP-based policy obtained from an optimal solution Single-OPT. The following theorem states that Single-OPT is an optimal policy for $\mathcal{P}((\lambda i r_{i})_{i \in I}, \epsilon)$.

**Theorem 4 (Optimality of Single-OPT).** Given an optimal solution $(\Phi^*, \pi^*, (u_i)_{i \in I})$ to the linear program $\mathcal{LP}((\lambda_i)_{i \in I}, \epsilon)$, we can solve the single-server optimization problem $\mathcal{P}((\lambda_i r_{i})_{i \in I}, \epsilon)$ in (3) with the optimal value $r / \Phi^*$, the optimal policy $\pi^*(\pi^*, (u_i)_{i \in I})$, and the optimal stationary distribution $\pi^*$. Moreover, the policy $\pi^*(\pi^*, (u_i)_{i \in I})$ is a Markovian policy.

**Proof of the Theorem 1 Based on Theorem 2, 3, and 4.**
Now we consider the single-server system. Let us start by introducing some useful notation. Let \( \sigma \) be the distribution of \( N(\sigma) \), then \((N(\sigma), \sigma, \pi)\) is a feasible solution to the problem \( P((\lambda_i r_i)_{i \in I}, \epsilon) \) in (3). As a result, we have \( N(\sigma) \geq N^* \).

The construction of the single-server policy \( \overline{\sigma} \) involves simulating an infinite-server system under \( \sigma \) from the empty configuration. At time 0, the policy \( \overline{\sigma} \) randomly chooses the \( \ell \)-th server in the infinite-server system with probability \( p^f \), for \( \ell = 1, 2, \ldots \). It then requests jobs for the single-server system according to a policy \( \overline{\sigma}^f \). The key to our policy \( \overline{\sigma}^f \) is to make the single-server system emulate the job assignment at the \( \ell \)-th server of the simulated infinite-server system, but without incurring idleness. We first construct the policy \( \overline{\sigma}^f \), and then specify the probabilities \( p^f \).

Let us start by introducing some useful notation. Let \( \overline{K}^f(t) \) be the single-server system configuration under \( \overline{\sigma}^f \) at time \( t \) and \( K^f(t) \) be the configuration of the \( \ell \)-th simulated server in the infinite-server system under \( \sigma \). We define a stochastic process \( \{s^f(t), t \geq 0\} \) as follows:

\[
s^f(t) = \max_{\tau} \{ s^f(t), t \geq 0 \}
\]

The “max” is well-defined because the integral is continuous in \( \tau \). Intuitively, \( s^f(t) \) gives the maximum time when the accumulative busy time of the \( \ell \)-th server is \( t \). Note that \( \{s^f(t), t \geq 0\} \) is only discontinuous when \( K^f(\tau) \) reaches 0, thus it is right-differentiable with derivative equal to 1 at any point.

We construct \( \overline{\sigma}^f \) and the simulation of the infinite-server system under \( \sigma \) in a way such that:

\[
\overline{K}^f(t) = K^f(s^f(t)) \quad \forall t.
\]

That is, we want that the single-server system has the same dynamic of the simulated \( \ell \)-th server except skipping the idle period. To this end, we couple the two systems as follows:

1. When the \( \ell \)-th simulated server \( K^f(s^f(t)) \) receives a type \( i \) job, we let the single-server system \( \overline{K}^f(t) \) request a type \( i \) job at time \( t \). For each such job, its phase transition process in the \( \ell \)-th simulated server is the same as that in the single-server system. That is, when we observe any internal transition or departure event in \( \overline{K}^f(t) \), we produce a same event on the \( \ell \)-th simulated server \( K^f(s^f(t)) \).

2. The simulations of the rest of the infinite-server system under policy \( \sigma \) are driven by independently generated random seeds.
It is not hard to see that the simulated infinite server-system has the same stochastic behavior as an uncoupled system under \( \sigma \). Moreover, as we couple all the events that happen in \( \bar{K}'(t) \) and \( K'_f(s'(t)) \), together with the facts that \( \bar{K}'(t) \) and \( K'_f(s'(t)) \) are piecewise constant and \( \bar{K}'(0-) = K'_f(s'(0)) = 0 \), we get (16).

Next we claim that (16) implies the following relationship between the steady-state cost of the single-server system under \( \bar{\sigma}' \) and the steady-state cost of the \( \ell \)-th simulated server in the infinite-server system under \( \sigma \):

\[
\mathbb{E} \left[ h(\bar{K}'(\infty)) \right] = \mathbb{E} \left[ h(K'_f(\infty)) \right],
\]

This is because for all \( k \neq 0 \), we have

\[
\frac{\mathbb{P} \left( K'_f(\infty) = k \right)}{\mathbb{P} \left( K'_f(\infty) \neq 0 \right)} = \lim_{S \to \infty} \frac{\int_0^S 1 \{ K'_f(s) = k \} ds}{\int_0^S 1 \{ K'_f(s) \neq 0 \} ds} = \lim_{T \to \infty} \frac{\int_0^T 1 \{ K'_f(s'(t)) = k \} dt}{\int_0^T 1 \{ K'_f(s'(t)) \neq 0 \} dt} = \lim_{T \to \infty} \frac{1}{T} \int_0^T 1 \{ K'_f(s'(t)) = k \} dt = \frac{1}{T} \int_0^T 1 \{ K'_f(s'(t)) = k \} dt \leq \mathbb{P} \left( \bar{K}'(\infty) = k \right),
\]

where (a) and (e) hold because long-run averages converge to steady-state expectations; (b) is due to the fact that \( \int_0^T 1 \{ K'_f(s'(t)) = k \} dt = \int_0^T 1 \{ K'_f(s) = k \} ds \), for any \( k \neq 0 \); (c) is due to the fact that \( \int_1^{1+} 1 \{ K'_f(s'(t)) \neq 0 \} dt \) and (d) follows from (16).

Let \( \bar{\lambda}'_i \) be the long-run request rates of type \( i \) jobs in the single-server system under \( \bar{\sigma}' \), and \( \lambda'_f \) be the throughput of type \( i \) jobs of \( \ell \)-th simulated server under \( \sigma \). By the construction of \( \bar{\sigma}' \), the single-server requests jobs based on the arrival events of the \( \ell \)-th simulated server, we have

\[
\bar{\lambda}'_i = \frac{\lambda'_f}{\mathbb{P}(K'_f(\infty) \neq 0)}, \quad \forall i \in I.
\]

With the constructed policies \( \{ \bar{\sigma}', \ell = 1, 2, \ldots \} \), we are ready to define the policy \( \bar{\sigma} \). We let \( \bar{\sigma} \) choose an index \( \ell \) with probability \( p'_\ell \) at time 0, and then follow \( \bar{\sigma}' \). We set the probability \( p'_\ell \) as

\[
p'_\ell = \frac{\mathbb{P} \left( K'_f(\infty) \neq 0 \right)}{\sum_{\ell=1}^{\infty} \mathbb{P} \left( K'_f(\infty) \neq 0 \right)} = \frac{\mathbb{P} \left( K'_f(\infty) \neq 0 \right)}{\sum_{k \neq 0} \mathbb{E}[X_k(\infty)]}, \quad \forall \ell = 1, 2, \ldots
\]

where the second inequality uses the fact that \( \sum_{\ell=1}^{\infty} \mathbb{P}(K'_f(\infty) \neq 0) = \sum_{k \neq 0} \mathbb{E}[X_k(\infty)] \). Then under \( \bar{\sigma} \), we have

\[
\mathbb{E} \left[ h(\bar{K}(\infty)) \right] = \sum_{\ell=1}^{\infty} p'_\ell \mathbb{E} \left[ h(\bar{K}'(\infty)) \right] = \sum_{k \neq 0} \mathbb{E}[X_k(\infty)] \cdot \frac{\sum_{\ell=1}^{\infty} \mathbb{P} \left( K'_f(\infty) \neq 0 \right)}{\sum_{\ell=1}^{\infty} \mathbb{P} \left( K'_f(\infty) \neq 0 \right)} \cdot \frac{\mathbb{E} \left[ h(K'_f(\infty)) \right]}{\mathbb{P} \left( K'_f(\infty) \neq 0 \right)} = \sum_{k \neq 0} h(k) \mathbb{E}[X_k(\infty)] = C(\sigma),
\]

which proves (14). Moreover, for each \( i \in I \) the request rate \( \bar{\lambda}'_i \) is given by

\[
\bar{\lambda}'_i = \sum_{\ell=1}^{\infty} p'_\ell \cdot \lambda'_f = \sum_{\ell=1}^{\infty} \mathbb{P} \left( K'_f(\infty) \neq 0 \right) \cdot \frac{\lambda'_f}{\mathbb{P}(K'_f(\infty) \neq 0)} = \frac{\sum_{\ell=1}^{\infty} \lambda'_f}{\sum_{k \neq 0} \mathbb{E}[X_k(\infty)]} = \frac{\lambda'_f}{N(\sigma)}.
\]

This proves (15). By the argument presented at the beginning of the proof, we get \( N(\sigma) \geq \bar{N}^* \). \( \square \)
6 PROOF OF THEOREM 3 (CONVERSION THEOREM)

In this section, we first prove Theorem 3 in the special case where the subroutine \( \sigma \) for JRRS is irreducible. Specifically, we describe the JRRS algorithm with an irreducible subroutine \( \sigma \) in Section 6.1 and provide the proof outline in Section 6.2. In Section 6.3, we prove Theorem 3 in the general setting by reducing it to the special case.

6.1 JRRS Assuming Irreducibility

JRRS is a meta-policy that takes a feasible solution of (3) as input. Specifically, the input consists of a Markovian single-server policy \( \sigma \), a stationary distribution \( \pi \) under \( \sigma \), and an objective value \( \bar{N} \). In this section, we focus on a slightly simplified version of the JRRS policy, assuming that the single-server policy \( \sigma \) is \( k^0 \)-irreducible, which is defined as follows.

**Definition 1.** A single-server policy is called \( k^0 \)-irreducible if under the policy there exists a configuration \( k^0 \) reachable from any other configurations through transitions.

**Preliminaries.** Let \( L = \lceil \bar{N} \rceil \). We divide the infinite server pool into two sets based on the server index \( t \): servers with index \( t \leq L \) are called normal servers, which serve most of the jobs; servers with index \( t > L \) are called backup servers, which are empty most of the time. We also define the parameter of token limit as \( \eta_{\text{max}} = \lceil L^{0.5} \rceil \).

A Markovian single-server policy makes the job-request decision based on the current server configuration \( k \). More specifically, the policy can request jobs either immediately after an event of internal transition or departure, or between two such events. We call a request happening right after an event a reactive request, and a request between two events a proactive request. When an event of internal transition or departure occurs, a Markovian policy can either make a reactive request immediately according to specified probability, or run Poisson clocks with specified rates for making a proactive request. When the policy decides to make a request, it produces a vector \( a \equiv (a_i)_{i \in I} \) that specifies the number of type \( i \) jobs to request for each \( i \in I \). All those decisions are based on the current configuration \( k \).

**Algorithm Description.** We let each normal server request jobs using the single-server policy \( \sigma \). Specifically, each normal server provides its configuration \( k \) to \( \sigma \), and informs \( \sigma \) when there is a departure or an internal transition event on the server. When the policy \( \sigma \) requests \( a = (a_i)_{i \in I} \) jobs, \( a_i \) type \( i \) tokens for each \( i \in I \) will be generated to track the requests, except when there are already some tokens on the server, in which case the requests will be ignored.

The tokens are used to guide the dispatch of arrivals. When a type \( i \) job arrives, the policy randomly chooses a type \( i \) token if there is one, and assigns the arrival to the corresponding server, and removes the token; when there are no type \( i \) tokens, the arrival is assigned to a backup server with the lowest index among servers that are not full. When the total number of type \( i \) tokens throughout the system exceeds the token limit \( \eta_{\text{max}} \), we generate a type \( i \) virtual arrival to check out a type \( i \) token, which brings a virtual job that has the same dynamics as a real job, but does not use resources. The server configuration for job requests includes both real jobs and virtual jobs.

Next we provide an intuition for the following fact: under JRRS, the steady-state real-job configuration of each normal server closely approximates that of a single-server system under \( \sigma \), and the backup servers are approximately empty. First, we argue that each token is turned into a job within a short period of time on the order of \( \Theta \left( r^{-0.5} \right) \). Observe that when the arrival rate is of order \( \Theta \left( r \right) \), the token limit is \( \eta_{\text{max}} = \Theta \left( r^{0.5} \right) \) and there are \( L = \Theta \left( r \right) \) normal servers. For each \( i \in I \), type \( i \) tokens are generated at a rate proportional to \( L \). Therefore, by Little’s Law, a type \( i \) token only exists for a time of order \( \eta_{\text{max}} / L = \Theta \left( r^{-0.5} \right) \) before turning into a real or virtual job. This implies that if we take into account of both real and virtual jobs, the configuration of a normal server
approximates that of a single-server system under $\sigma$. Moreover, we can show that the number of tokens rarely exceeds the token limit or reaches zero, so there are almost no virtual jobs or jobs on the backup servers. As a result, the real-job configuration in each normal server closely approximates that of a single-server system under $\sigma$, and the backup servers are almost empty.

Note that due to the way each normal server requests jobs, the total number of real jobs, virtual jobs, and tokens on a normal server never exceeds $K_{\text{max}}$. In particular, the single-server policy $\sigma$ requests jobs only when there are no tokens on the server, and it will not request more than $K_{\text{max}} - n$ jobs if there are already $n$ real and virtual jobs on the server.

6.2 Proof Overview

Preliminaries. Before presenting the proof outline, we first introduce some notation. Consider an infinite-server system under the JRRS policy. For each normal server $\ell$, we describe its configuration at time $t$ using the following variables: configuration of real jobs $K^\ell(t)$, tokens $\eta^\ell(t)$, configuration of virtual jobs $\xi^\ell(t)$, and observed configuration $\tilde{K}^\ell(t) \triangleq K^\ell(t) + \xi^\ell(t)$. We use the superscript "1: $L$" to refer to a certain descriptor of all normal servers, for example, $\tilde{K}^{1:L}(t) \triangleq (\tilde{K}^\ell(t))_{\ell=1,2,\ldots}$. The system under the JRRS policy is a Markov chain with a unique Markovian representation $((K^\ell(t))_{\ell=1,2,\ldots}, \xi^{1:L}(t), \eta^{1:L}(t))$. The following lemma shows that the system has a unique stationary distribution (the proof is provided in Appendix A.1).

Lemma 1 (Unique Stationary Distribution). Consider an infinite-server system under the JRRS policy with $\sigma$ as its subroutine, where $\sigma$ is a single-server policy that is Markovian and $K^{1:-}$-irreducible. Then the state of the system $((K^\ell(t))_{\ell=1,2,\ldots}, \xi^{1:L}(t), \eta^{1:L}(t))$ has a unique stationary distribution.

Let $\overline{K}^{1:L}(t) \triangleq (\overline{K}^\ell(t))_{\ell=1,2,\ldots,L}$ be the configuration of $L$ i.i.d. copies of the single-server system under $\sigma$. As discussed in Section 6.1, we will show that $K^{1:L}(\infty)$ can be approximated by $\overline{K}^{1:L}(\infty)$.

To rigorously discuss the approximation of the steady-state random variables, we define some metrics. Recall that $\mathcal{K} \triangleq \{k: \sum_{i \in I} k_i \leq K_{\text{max}}\}$ is the set of feasible single-server configuration. Let $\mathcal{K}^L \triangleq \{k^{1:L}: k^\ell \in \mathcal{K}, \forall \ell\}$ be the set of feasible configuration of all normal servers. We use $||\cdot||$ to denote the $L^1$ norm in both space $\mathcal{K}$ and space $\mathcal{K}^L$:

$$||k - k'|| = \sum_{i \in I} |k_i - k'_i|,$$

for $k, k' \in \mathcal{K}$,

$$||k^{1:L} - k'^{1:L}|| = \sum_{\ell=1}^L ||k^\ell - k'^\ell||,$$

for $k^{1:L}, k'^{1:L} \in \mathcal{K}^L$.

For any two random variables $U^a, U^b \in \mathcal{K}^L$, their closeness will be measured in terms of Wasserstein distance as follows:

$$d(U^a, U^b) \triangleq \sup_{f \in \text{Lip}(1)} \left\{ \mathbb{E} \left[ f \left( U^a \right) \right] - \mathbb{E} \left[ f \left( U^b \right) \right] \right\},$$

where the supremum is taken over the function class $\text{Lip}(1)$

$$\text{Lip}(1) \triangleq \{ \text{function } f \text{ on } \mathcal{K}^L: \left| f(k^{1:L}) - f(k'^{1:L}) \right| \leq ||k^{1:L} - k'^{1:L}|| \}.$$

In the remaining of this section, we omit the steady-state symbol $(\infty)$ when they are clear from the context.

Proof Outline. Observe that the process $((\tilde{K}^{1:L}(t), \eta^{1:L}(t)))$ itself forms a Markov chain and governs the dynamics of other state variables, including the number of virtual jobs $\xi^{1:L}(t)$ and the number of jobs on the backup servers. We can thus decompose our analysis into two steps. In Step 1, we focus on the dynamics of $((\tilde{K}^{1:L}(t), \eta^{1:L}(t)))$ and prove that $\tilde{K}^{1:L}$ is close to $\overline{K}^{1:L}$ in terms of the Wasserstein distance. In Step 2, we use the result of the first step to show that the total number of virtual jobs $\sum_{i \in I} \sum_{\ell=1}^L \xi^\ell_i$ and the total number of jobs on the backup servers are both small. Recall
that the real job configuration is given by $K^{1:L} = \hat{K}^{1:L} - \zeta^{1:L}$. **Step 1** and **Step 2** together imply that $K^{1:L}$ and $\hat{K}^{1:L}$ are close in the Wasserstein distance, and the backup servers are almost empty. Since $\hat{K}^{1:L}$ are i.i.d. copies of the single-server system, we thus establish a connection between the configurations of the infinite-server system $(K^f)_{f=1,2,\ldots}$ and the stationary distribution of the single-server system. Then the theorem follows.

**Step 1.** We prove Lemma 2, which bounds the Wasserstein distance between $\hat{K}^{1:L}$ and $\tilde{K}^{1:L}$.

**Lemma 2.** Under the conditions of Theorem 3 and $\tilde{\sigma}$ being $k^0$-irreducible, we have

$$d\left(\tilde{K}^{1:L}, \hat{K}^{1:L}\right) = O\left(r^{0.5}\right).$$

The proof of Lemma 2 follows the framework of Stein’s method (see, for example [6–8]). Stein’s method usually consists of three steps: generator comparison, Stein factor bounds, and moment bounds. In our case, due to the finiteness of the state space $\mathcal{K}$, we only need to do the generator comparison and the Stein factor bounds. The generator comparison step justifies the intuition that most normal servers behave like single-server systems under the policy $\tilde{\sigma}$, and the Stein’s factor bounds corresponds to the intuition that the small fraction of servers whose dynamics deviate from single-server systems do not contribute much to the overall distance. We defer the detailed proof to Appendix A.2.

**Step 2.** We establish the following Lemma 3 and Lemma 4, which bounds the steady-state expected number of virtual jobs and the number of jobs on backup servers, respectively.

**Lemma 3.** Under the conditions of Theorem 3 and $\tilde{\sigma}$ being $k^0$-irreducible, for each $i \in I$, the steady-state expected number of virtual jobs of type $i$ is of the order $O\left(r^{0.5}\right)$, i.e.

$$E\left[\sum_{\ell=1}^L \zeta^f_{\ell, i}\right] = O\left(r^{0.5}\right).$$

**Lemma 4.** Under the conditions of Theorem 3 and $\tilde{\sigma}$ being $k^0$-irreducible, for each $i \in I$, the steady-state expected number of type $i$ jobs on backup servers is of the order $O\left(r^{0.5}\right)$, i.e.

$$E\left[\sum_{\ell=L+1}^\infty K^f_{\ell, i}\right] = O\left(r^{0.5}\right).$$

The proof of Lemmas 3 and 4 is provided in Appendix A.3. Theorem 3 with $\tilde{\sigma}$ being $k^0$-irreducible then follows from the three lemmas above.

**Proof of Theorem 3 with $k^0$-irreducible $\tilde{\sigma}$.** First we show that Lemma 2 and 3 imply the closeness between $K^{1:L}$ and $\hat{K}^{1:L}$. From the Wasserstein distance bound in Lemma 2, we have

$$E\left[f(K^{1:L})\right] - E\left[f(\hat{K}^{1:L})\right] = O\left(r^{0.5}\right),$$

for any $f : \mathcal{K}^L \to \mathbb{R}$ that is 1-Lipschitz continuous. Recall that $\hat{K}^f = K^f + \zeta^f$, and $E\left[\sum_{\ell=1}^L \zeta^f_{\ell, i}\right] = O\left(r^{0.5}\right)$ by Lemma 3. Therefore, by the Lipschitz continuity of $f$, we have

$$E\left[f(K^{1:L})\right] - E\left[f(\hat{K}^{1:L})\right] = O\left(r^{0.5}\right).$$

Now we prove (6) by leveraging the Wasserstein distance bound above. We write the difference in the expected number of active servers between the two systems as follows:

$$\sum_{k \neq 0} E\left[X_k\right] - L \cdot P(\mathcal{K} \neq 0) = E\left[\sum_{\ell=1}^L \mathbf{1}_{(K^f_{\ell, i} \neq 0)}\right] + E\left[\sum_{\ell=L+1}^\infty \mathbf{1}_{(K^f_{\ell, i} \neq 0)}\right] - E\left[\sum_{\ell=1}^L \mathbf{1}_{(\hat{K}^f_{\ell, i} \neq 0)}\right]

= E\left[f_1(K^{1:L})\right] - E\left[f_1(\hat{K}^{1:L})\right] + E\left[\sum_{\ell=L+1}^\infty \mathbf{1}_{(\hat{K}^f_{\ell, i} \neq 0)}\right],$$

(23)
where \( f_1 \) is given by \( f_1(k^{1:L}) = \sum_{l=1}^{L} \mathbb{1}_{\{k^l \neq 0\}} \). Note that for any \( k^{1:L}, k'^{1:L} \in \mathcal{K}^L \), we have
\[
|f_1(k^{1:L}) - f_1(k'^{1:L})| = \left| \sum_{l=1}^{L} \mathbb{1}_{\{k^l \neq 0\}} - \mathbb{1}_{\{k'^l \neq 0\}} \right| \leq \sum_{l=1}^{L} \mathbb{1}_{\{k^l \neq k'^l\}} \leq \|k^{1:L} - k'^{1:L}\|.
\]
That is, \( f_1 \in \text{Lip}(1) \). Applying (22) and Lemma 4 to (23), we get
\[
\mathbb{E} \left[ \sum_{k \neq 0} X_k \right] - L \cdot \mathbb{P}(K \neq 0) = O \left( r^{0.5} \right).
\]

Similarly, to prove (7), we write the difference in cost between the two systems as follows:
\[
\sum_{k \neq 0} h(k) \mathbb{E}[X_k] - L \cdot \mathbb{E}[h(K)] = \mathbb{E} \left[ \sum_{l=1}^{L} h(K^l) \right] + \mathbb{E} \left[ \sum_{l=L+1}^{\infty} h(K^l) \right] = \mathbb{E} \left[ \sum_{l=1}^{L} h(K^l) \right] + \mathbb{E} \left[ \sum_{l=L+1}^{\infty} h(K^l) \right] \cdot \max_{k \in \mathcal{K}} h(k),
\]
where \( \Gamma \) is the Lipschitz constant of \( h(\cdot) \), and \( f_2 \) is given by \( f_2(k^{1:L}) = \frac{1}{L} \sum_{l=1}^{L} h(k^l) \). For any \( k^{1:L}, k'^{1:L} \in \mathcal{K}^L \), we have
\[
|f_2(k^{1:L}) - f_2(k'^{1:L})| = \left| \sum_{l=1}^{L} (h(k^l) - h(k'^l)) \right| \leq \sum_{l=1}^{L} \|k^l - k'^l\| = \|k^{1:L} - k'^{1:L}\|,
\]
i.e., \( f_2 \in \text{Lip}(1) \). Applying (22) and Lemma 4 to (23), we get
\[
\sum_{k \neq 0} h(k) \mathbb{E}[X_k] - L \cdot \mathbb{E}[h(K)] = O \left( r^{0.5} \right).
\]
To avoid repetition, we leave the derivation of (8) and (9) to the proof of the general case below. \( \square \)

### 6.3 Proof of Conversion Theorem without Assuming Irreducibility

In this subsection, we prove Theorem 3 without assuming \( k^0 \)-irreducibility of the subroutine \( \bar{\sigma} \). Specifically, suppose that we have a Markovian single-server policy \( \bar{\sigma} \) and an initial distribution \( p_j \) over its recurrent classes \( S_j \) for \( j = 1, 2, \ldots J \), we will construct an infinite-server policy \( \sigma \) such that (6)- (7) still hold.

We first state a basic fact about recurrent classes: if there are at most \( m \) jobs in any configuration of a recurrent class \( S \), then any configurations with \( m \) jobs must be in \( S \).

**Lemma 5 (Maximal Layer of a Recurrent Class).** Suppose \( S \) is a recurrent class under a certain policy \( \bar{\sigma} \). Define the maximal layer of \( S \) as the set \( \text{ML}(S) \triangleq \{ k \in \mathcal{K} | \sum_i k_i = \max_{k' \in S} \sum_i k'_i \} \). Then we have
\[
\text{ML}(S) \subseteq S. \tag{24}
\]

Based on Lemma 5, we can break a general Markovian single-server policy \( \bar{\sigma} \) into many \( k^0 \)-irreducible Markovian policies, each induces one recurrent class and preserves stationary distribution and the throughput of \( \bar{\sigma} \) on that recurrent class.

**Lemma 6 (Breaking The Reducible Policy).** Let \( \bar{\sigma} \) be a general single-server Markovian policy with recurrent classes \( S_j \) for \( j = 1, 2, \ldots, J \). Then for each \( j \) exists a Markovian policy \( \bar{\sigma}^j \) such that
- The induced Markov chain is \( k^0 \)-irreducible with the unique recurrent class being \( S_j \);
- The stationary distribution is the same as the stationary distribution under \( \bar{\sigma} \) starting from a configuration in \( S_j \).

We provide the proof of Lemma 5 and Lemma 6 in Appendix A.4 and A.5, respectively.

For \( j = 1, \ldots, J \), let \( \pi^j \) and \( \bar{\pi}^j \)’s be the stationary distribution and throughput of the policy \( \bar{\sigma}^j \).

By Lemma 6, we have the following relationships:
\[
\pi(k) = \sum_{j=1}^{J} \pi^j(k), \quad \forall k \in \mathcal{K}, \tag{25}
\]
Based on the above relationships, we can prove the general version of Conversion Theorem that does not require \( k^0 \)-irreducibility.

**Proof of Theorem 3.** For each \( j = 1, \ldots, J \), Lemma 6 implies that the policy \( \sigma^j \) and stationary distribution \( \pi^j \) form a feasible solution to the single-server optimization problem \( \bar{P}(\{p^j/\lambda^j_i\}_{i \in I}, \varepsilon) \), and the corresponding objective value is \( p^j/\bar{N} \). Consider the infinite-server system with arrival rates \( \{p^j/\lambda^j_i\}_{i \in I} \) and budget \( \varepsilon \). As we have proved Theorem 3 for the JRRS policy with \( k^0 \)-irreducible subroutine \( \sigma^j \), it follows that

\[
\sum_{k \neq 0} E[X_k^j] - \left[ p^j/\bar{N} \right] \cdot (1 - \pi^j(0)) = O(r^{0.5}),
\]

\[
\sum_{k \neq 0} h(k)E[X_k^j] - \left[ p^j/\bar{N} \right] \cdot \sum_{k \neq 0} h(k)\pi^j(k) = O(r^{0.5}),
\]

where \( X_k^j \) is the random variable representing the steady-state number of servers in configuration \( k \) in the infinite-server system under the JRRS policy \( \sigma^j \) with subroutine \( \sigma^j \).

Now we construct the policy \( \sigma \) that works for an infinite-server system with arrival rates \( \lambda_i r \) for each \( j = 1, \ldots, J \). Then the arrival rate to the \( j \)-th infinite-server system is equal to \( \sum_j p^j/\lambda_i \cdot \lambda_i r = \sum_j p^j/\lambda_i \cdot \lambda_i r = p^j/\lambda_i \cdot \bar{N} \), where the first equality is due to (26), and the second equality is due to the condition that \( \lambda_i r = \lambda_j^j \cdot L \). Therefore, (27) and (28) hold, and we have

\[
\left| \sum_{k \neq 0} E[X_k] - \bar{N} \cdot (1 - \pi(0)) \right| = \sum_{j=1}^J \sum_{k \neq 0} E[X_k^j] - \sum_{j=1}^J p^j/\bar{N} \cdot (1 - \pi^j(0)) \leq \sum_{j=1}^J \sum_{k \neq 0} E[X_k^j] - \left[ p^j/\bar{N} \right] \cdot (1 - \pi^j(0)) + O(1) = O(r^{0.5}).
\]

Here we use (27) and the relationship between \( \pi(k) \) and \( \pi^j(k) \). Similarly,

\[
\left| \sum_{k \neq 0} h(k)E[X_k] - \bar{N} \cdot \sum_{j=1}^J \sum_{k \neq 0} h(k)\pi(k) \right| = \sum_{j=1}^J \sum_{k \neq 0} E[X_k^j] - \sum_{j=1}^J p^j/\bar{N} \cdot \sum_{k \neq 0} h(k)\pi^j(k)
\]

\[
= \sum_{j=1}^J \sum_{k \neq 0} E[X_k^j] - \left[ p^j/\bar{N} \right] \cdot \sum_{k \neq 0} h(k)\pi^j(k) + O(1) = O(r^{0.5}).
\]

After re-indexing the servers, we get (6) and (7).

To show (8) and (9), noting that \( [\bar{N}] = \Theta(r) \), we have

\[
N(\sigma) = \sum_{k \neq 0} E[X_k] \leq \bar{N} + O(r^{0.5}) = (1 + O(r^{-0.5})) \cdot \bar{N},
\]

and

\[
C(\sigma) = \frac{\sum_{k \neq 0} h(k)E[X_k]}{\sum_{k \neq 0} E[X_k^j]} = \frac{\sum_{j=1}^J \sum_{k \neq 0} h(k)\pi(k) + O(r^{-0.5})}{1 - \pi(0) + O(r^{-0.5})}
\]

\[
= (1 + O(r^{-0.5})) \cdot E[h(\bar{K}(\infty))|\bar{K}(\infty) \neq 0] \leq (1 + O(r^{-0.5})) \cdot \varepsilon.
\]
This finishes the proof.

7 ANALYSIS OF SINGLE-SERVER SYSTEM

In this section, we solve the single-server optimization problem in (3) through the linear program (LP) in (10). We first derive (10) as a linear program relaxation of the single-server optimization problem, and then construct a single-server policy that achieves the optimal value of the LP, which implies the optimality of the policy. Moreover, the constructed single-server policy is a Markovian policy, so it can be converted to an asymptotically optimal policy for the infinite-server optimization problem through JRRS.

7.1 Lower Bound via LP Relaxation

In this subsection we derive an LP relaxation of the optimization problem (3), restated below:

$$\begin{align*}
\text{minimize} & \quad \bar{N} \\
\text{subject to} & \quad \mathbb{E} \left[ h(\bar{K}(\infty) \mid \bar{K}(\infty) \neq 0) \right] \leq \epsilon, \\
& \quad \bar{N} \cdot \bar{\lambda}_i = \lambda_i r, \quad \forall i \in I.
\end{align*}$$

(3 revisited)

Observe that both $\bar{K}(\infty)$ and $\bar{\lambda}_i$ depend on the stationary distribution $\pi$, but the constraints in terms of $\pi$ are implicit. To derive an LP relaxation, we give an explicit characterization of the constraints that must be satisfied by the stationary distribution $\pi$ induced by any feasible policy $\sigma$.

To do this, we derive a version of the stationary equation in terms of a quantity called transition frequency. The transition frequency of type $i$ jobs is a function $u_i : \mathcal{K} \to \mathbb{R}$ describing the steady-state frequency of requesting a type $i$ job when the system has configuration $k$. To rigorously define transition frequency, we first introduce a concept called nominal transition.

**Definition 2 (Nominal Transition).** Consider a single-server system under any policy. When the configuration $\bar{K}(t)$ transitions from $k$ to $k' + a$ for some $k, k' \in \mathcal{K}$ with $a = (a_i)_{i \in I}$ new jobs added into service, we decompose the transition by adding intermediate configurations as illustrated below, where $k$ first goes to $k'$ if $k' \neq k$, then add jobs of each type one by one.

$$
k \rightarrow k' \rightarrow (k' + e_i) \rightarrow \cdots \rightarrow (k' + a_i e_i) \rightarrow \cdots \rightarrow (k' + a_i e_i + \cdots + a_i |I| e_i),
$$

where $(i_1, i_2, \ldots, i_{|I|})$ is a fixed ordering of the set of phases $I$. We call each short transition in the diagram a nominal transition.

For $k^1, k^2 \in \mathcal{K}$, we denote $F(k^1, k^2, t)$ as the cumulative number of nominal transitions from $k^1$ to $k^2$ during the time interval $[0, t]$, which is a random variable with a distribution depending on the single-server policy and initial distribution of configurations.

Note that for any $i \in I$ and $k \in \mathcal{K}$ s.t. $k_i \geq 1$, $F(k, k - e_i, t)$ counts the number of times that a type $i$ job departs when being in configuration $k$. As a result,

$$F(k, k - e_i, t) = \mathcal{N} \left( \int_0^t k_i \mu_{i\perp} \mathbb{1}_{\{K(s) = k\}} ds \right),$$

where $\mathcal{N}(t)$ denotes a unit rate Poisson process. If we take expectation, divide both sides by $t$, and let $t \to \infty$, we have

$$
\lim_{t \to \infty} \frac{1}{t} \mathbb{E}[F(k, k - e_i, t)] = k_i \mu_{i\perp} \cdot \lim_{t \to \infty} \frac{1}{t} \int_0^t \mathbb{P}(K(s) = k) ds = k_i \mu_{i\perp} \pi(k).
$$

(29)
Similarly, \( F(k, k - e_i + e_{i'}, t) \) counts the number of times a job in phase \( i \) transitions to phase \( i' \) when being in configuration \( k \) for any \( i, i' \in \mathcal{I}, k \in \mathcal{K} \) s.t. \( i' \neq i \) and \( k_i \geq 1 \), so
\[
\lim_{t \to \infty} \frac{1}{t} \mathbb{E}[F(k, k - e_i + e_{i'}, t)] = \kappa_i \mu_i \pi(k).
\] (30)

We define transition frequency as follows.

**Definition 3** (Transition Frequency). Transition frequency of type \( i \) jobs at state \( k \) is the long-run average number of nominal transitions from configuration \( k \) to \( k + e_i \) per unit time,
\[
u_i(k) \doteq \lim_{t \to \infty} \frac{1}{t} \mathbb{E}[F(k, k + e_i, t)].
\] (31)

The transition frequencies allow us to derive the following version of stationary equation.

**Lemma 7** (Stationary Equation). Under any policy, the stationary distribution \( \pi \) and the transition frequency \( \nu_i \) satisfy the following equation:
\[
\sum_i \nu_i(k - e_i) \mathbb{I}_{\{k_i \geq 1\}} + \sum_i (k_i + 1) \mu_i \pi(k + e_i) + \sum_{i,i' : i \neq i'} (k_i + 1) \mu_i \pi(k + e_i - e_{i'}) \mathbb{I}_{\{k_{i'} \geq 1\}}
\]
\[
= \sum_i \nu_i(k) + \left( \sum_i k_i \mu_i + \sum_{i,i' : i \neq i'} k_i \mu_i \mathbb{I}_{\{k_i \geq 1\}} \right) \pi(k)
\] (32)

for any state \( k \in \mathcal{K} \), and \( \sum_i, \sum_{i,i'} \) are shorthand for \( \sum_{i \in \mathcal{I}}, \sum_{i,i' \in \mathcal{I}} \).

**Proof.** For each configuration \( k \in \mathcal{K} \), if we look at difference between the number of nominal transitions into configuration \( k \) and that out of configuration \( k \) by time \( t \), we have the following equation,
\[
\mathbb{I}_{\{X(t) = k\}} - \mathbb{I}_{\{X(0) = k\}}
\]
\[
= \sum_i \mathbb{I}(F(k, k - e_i, t) \mathbb{I}_{\{k_i \geq 1\}}) + \sum_i \mathbb{I}(F(k, k + e_i, t)) + \sum_{i,i' : i \neq i'} \mathbb{I}(F(k, k + e_i, t) \mathbb{I}_{\{k_{i'} \geq 1\}})
\]
\[
- \sum_i \mathbb{I}(F(k, k - e_i, t)) - \sum_i \mathbb{I}(F(k, k + e_i, t) \mathbb{I}_{\{k_i \geq 1\}}) - \sum_{i,i' : i \neq i'} \mathbb{I}(F(k, k + e_i, t) \mathbb{I}_{\{k_{i' \geq 1\}})}
\] (33)

By Definition 3 and (29)-(30), if we divide both sides of (33) by \( t \) and let \( t \to \infty \), we get the stationary equation in (32).

Since the stationary equation in (32) is linear in \( \nu_i(k) \) and \( \pi(k) \), we can write it in matrix form:
\[
A \pi + \sum_{i \in \mathcal{I}} B_i u_i = 0,
\] (34)
where \( \pi \) and \( u_i \) are column vectors representing \( \pi(\cdot) \), \( u_i(\cdot) \), and \( A, B_i \) are matrices that make (34) equivalent to (32). Therefore, the following three conditions are necessary for any tuple \( (\pi, (u_i)_{i \in \mathcal{I}}) \) to be a possible pair of stationary distribution and transition frequencies for a Markovian policy.
\[
A \pi + \sum_{i \in \mathcal{I}} B_i u_i = 0
\]
\[
\sum_k \pi(k) = 1
\]
\[
\pi, u_i \geq 0, \quad \forall i \in \mathcal{I}
\] (35)

Now we can convert the single-server control problem \( \mathcal{P}((\lambda_i)_{i \in \mathcal{I}}, \epsilon) \) in (3) to the linear program \( \mathcal{P}((\lambda_i)_{i \in \mathcal{I}}, \epsilon) \) defined in (10).
Note that (34) is the necessary condition for any policy, thus any feasible solution to (3) is also feasible to (10). Let \( \hat{N} \) be the optimal value of (3) and \( \Phi^* \) be the optimal value of (10). We have
\[
\hat{N} \geq \frac{r}{\Phi^*}.
\]

### 7.2 Policy Construction

In this subsection, we describe a procedure that allows us to construct a policy that achieves the lower bound given by the LP relaxation in (10). Specifically, given a feasible solution \((\pi, (u_i)_{i \in I})\) to (10), we define an LP-based policy that requests jobs as follows:

- **Case 1**: If the current configuration is \(k\) with \(\pi(k) \neq 0\), the policy runs a Poisson clock with rate \(\frac{u_i(k)}{\pi(k)}\) for each \(i \in I\), and requests a type \(i\) job when the corresponding clock ticks;
- **Case 2**: If after a departure or an internal transition, or after a new job is requested, the configuration is \(k\) with \(\pi(k) = 0\) and \(\sum_{i' \in I} u_{i'}(k) \neq 0\), the policy immediately requests a type \(i\) job with probability \(\frac{u_i(k)}{\sum_{i' \in I} u_{i'}(k)}\);
- **Case 3**: If after a departure or an internal transition, or after a new job is requested, the configuration is \(k\) with \(\pi(k) = 0\), \(\sum_{i' \in I} u_{i'}(k) = 0\), and \(\sum_{i \in I} k_i < \max\{\sum_{i \in I} k_i' : \pi(k') \neq 0\}\), the policy immediately requests a type \(i\) job with uniform probability.

We denote the LP-based policy based on the solution \((\pi, (u_i)_{i \in I})\) as \(\bar{\sigma}(\pi, (u_i)_{i \in I})\). When an LP-based policy is used as a subroutine of the JRRS policy described in Section 6.1, its proactive requests are always initiated by an action in Case 1, possibly followed by several actions in Case 2 and Case 3; its reactive requests are initiated by a departure or internal transition and followed by actions in Case 2 or Case 3. Note that although it seems that each case above only requests one job at a time, because Case 2 and Case 3 requires “immediate” actions, multiple jobs could be requested in an instant and are regarded as one proactive or reactive request.

The following lemma characterizes the steady-state behavior of a single-server system under an LP-based policy.

**Lemma 8** (Properties of LP-based Policies). Consider a single-server system under the LP-based policy \(\bar{\sigma}(\pi, (u_i)_{i \in I})\), where \((\pi, (u_i)_{i \in I})\) is a feasible solution to (10). We have that \(\pi\) is a stationary distribution under policy \(\bar{\sigma}\), and \((u_i)_{i \in I}\) are the transition frequencies corresponding to \(\bar{\sigma}\).

The proof of Lemma 8 is based on (33), following the same argument as the proof of Lemma 7, as well as an induction argument. We defer the full proof to Appendix B.1 due to the space limit.

Theorem 4 then follows immediately from Lemma 8. The proof is provided in Appendix B.2.

### REFERENCES

[1] Noman Bashir, Nan Deng, Krzysztof Rzadca, David Irwin, Sree Kodak, and Rohit Jnagal. 2021. Take It to the Limit: Peak Prediction-Driven Resource Overcommitment in Datacenters. In Proc. European Conf. Computer Systems (EuroSys). Online Event, United Kingdom, 556–573.

[2] Benjamin Berg, Jan-Pieter Dorsman, and Mor Harchol-Balter. 2017. Towards Optimality in Parallel Scheduling. Proc. ACM Meas. Anal. Comput. Syst. 1, 2, Article 40 (Dec. 2017), 30 pages.

[3] Benjamin Berg, Mor Harchol-Balter, Benjamin Moseley, Weina Wang, and Justin Whitehouse. 2020. Optimal Resource Allocation for Elastic and Inelastic Jobs. In Proc. Ann. ACM Symp. Parallelism in Algorithms and Architectures (SPAA). Virtual Event, USA, 75–87.

[4] Benjamin Berg, Rein Vesilo, and Mor Harchol-Balter. 2020. heSRPT: Parallel scheduling to minimize mean slowdown. Perform. Eval. 144 (2020), 102147.

[5] Benjamin Berg, Justin Whitehouse, Benjamin Moseley, Weina Wang, and Mor Harchol-Balter. 2022. The case for phase-aware scheduling of parallelizable jobs. Perform. Eval. 153 (2022), 102246.

[6] Anton Braverman. 2022. The Prelimit Generator Comparison Approach of Stein’s Method. Stoch. Syst. 12, 2 (2022), 181–204.
Time-varying Resource Requirements

[7] Anton Braverman and J. G. Dai. 2017. Stein’s method for steady-state diffusion approximations of $M/Ph/n + M$ systems. *Ann. Appl. Probab.* 27 (Feb. 2017), 550–581.

[8] Anton Braverman, J. G. Dai, and Jieckun Feng. 2017. Stein’s method for steady-state diffusion approximations: an introduction through the Erlang-A and Erlang-C models. *Stoch. Syst.* 6, 2 (2017), 301–366.

[9] Niv Buchbinder, Yaron Fairstein, Konstantina Mellou, Ishai Menache, and Joseph (Seffi) Naor. 2021. Online Virtual Machine Allocation with Lifetime and Load Predictions. *ACM SIGMETRICS Perform. Eval. Rev.* 49, 1 (May 2021), 9–10.

[10] Google Cloud. 2022. Overcommitting CPUs on sole-tenant VMs. https://cloud.google.com/compute/docs/nodes/overcommitting-cpus-sole-tenant-vms.

[11] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Salt Lake City, UT, 127–144.

[12] Sergey Foss and Alexander Stolyar. 2017. Large-scale Join-Idle-Queue system with general service times. *J. Appl. Probab.* 54, 4 (2017), 995–1007.

[13] Apache Software Foundation. 2022. Apache Mesos: Oversubscription. https://mesos.apache.org/documentation/latest/oversubscription/.

[14] Javad Ghaderi, Yuan Zhong, and R. Srikant. 2014. Asymptotic Optimality of BestFit for Stochastic Bin Packing. *SIGMETRICS Perform. Eval. Rev.* 42, 2 (Sept. 2014), 64–66.

[15] Diego Goldsztajn, Sem C. Borst, and Johan S. H. van Leeuwaarden. 2021. Utility maximizing load balancing policies. https://arxiv.org/abs/2112.08958.

[16] Diego Goldsztajn, Sem C. Borst, Johan S. H. van Leeuwaarden, Debankur Mukherjee, and Philip A. Whiting. 2022. Self-Learning Threshold-Based Load Balancing. *INFORMS J. Comput.* 34 (Jan. 2022), 39–54. Issue 1.

[17] Isaac Grosof, Mor Harchol-Balter, and Alan Scheller-Wolf. 2020. Stability for Two-class Multiserver-job Systems. https://arxiv.org/abs/2010.00631.

[18] Sönke Hartmann. 2013. Project scheduling with resource capacities and requests varying with time: A case study. *Flex. Serv. Manuf. J.* 25 (2013), 74–93. Issue 1-2.

[19] Sönke Hartmann. 2015. *Time-Varying Resource Requirements and Capacities*. Springer, Cham, 163–176.

[20] Sönke Hartmann and Dirk Briskorn. 2022. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* 297 (Feb. 2022), 1–14. Issue 1.

[21] Yige Hong and Weina Wang. 2022. Sharp Waiting-Time Bounds for Multiserver Jobs. In *ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*. Seoul, South Korea.

[22] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2015. Heracles: Improving resource efficiency at scale. In *Proc. ACM/IEEE Ann. Int. Symp. Computer Architecture (ISCA)*. Portland, OR, 450–462.

[23] Yi Lu, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R. Larus, and Albert Greenberg. 2011. Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services. *Perform. Eval.* 68, 11 (Nov. 2011), 1056–1071.

[24] Siva Theja Maguluri and R. Srikant. 2013. Scheduling jobs with unknown duration in clouds. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*. Turin, Italy, 1887–1895.

[25] Siva Theja Maguluri, R Srikant, and Lei Ying. 2012. Stochastic Models of Load Balancing and Scheduling in Cloud Computing Clusters. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*. Orlando, FL, 702–710.

[26] Sean Meyn. 2007. *Control Techniques for Complex Networks* (1st ed.). Cambridge University Press, USA.

[27] Debankur Mukherjee and Alexander Stolyar. 2019. Join Idle Queue with Service Elasticity: Large-Scale Asymptotics of a Nonmonotone System. *Stoch. Syst.* 9 (Dec. 2019), 338–358. Issue 4.

[28] Konstantinos Psychas and Javad Ghaderi. 2018. On Non-Preemptive VM Scheduling in the Cloud. In *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*. Association for Computing Machinery, Irvine, CA, 67–69.

[29] Konstantinos Psychas and Javad Ghaderi. 2019. Scheduling Jobs with Random Resource Requirements in Computing Clusters. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*. Paris, France, 2269–2277.

[30] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. 2012. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *Proc. ACM Symp. Cloud Computing (SoCC)*. San Jose, CA, Article 7, 13 pages.

[31] Alexander L. Stolyar. 2015. Tightness of Stationary Distributions of a Flexible-Server System in the Halfin-Whitt Asymptotic Regime. *Stoch. Syst.* 5, 2 (2015), 239–267.

[32] Alexander L. Stolyar. 2017. Large-scale heterogeneous service systems with general packing constraints. *Adv. Appl. Probab.* 49 (March 2017), 61–83. Issue 1.

[33] Alexander L. Stolyar. 2017. Pull-based load distribution among heterogeneous parallel servers: the case of multiple routers. *Queueing Syst.* 85 (Feb. 2017), 31–65. Issue 1-2.
We construct the Lyapunov function $g$ as follows:

$$g((k^t)_t = 1,2,...,\xi^{1:L}(t), \eta^{1:L}(t)) = \sum_{i \in I} \sum_{t=1}^{\infty} t_i k_i^t + \sum_{i \in I} \sum_{t=1}^{\infty} t_i \xi_i^t.$$  

Using the relation (38), it can be verified that the drift of $g$ satisfies

$$\tilde{G}g((k^t)_t = 1,2,...,\xi^{1:L}(t), \eta^{1:L}(t)) \leq \sum_{i \in I} \left( \lambda_i t_i r - \sum_{t=1}^{\infty} k_i^t \right)$$  

where $g$ is a non-negative function of the states, $S$ is a finite set, $b$ is a finite number, and $\tilde{G}$ is the infinitesimal generator of the continuous time Markov chain. Let $t_i$ be the expected time in system of type $i$ jobs for each $i \in I$. According to the job model, we have the relation

$$\left( \mu_{i} + \sum_{i' \in I : i' \neq i} \mu_{i'} \right) t_i = \sum_{i' \in I : i' \neq i} \mu_{i'} t_{i'} \quad \forall i \in I.$$  

We will show that under the JRRS policy the Markov chain $(K^t(t))_{t=1,2,...}$ is irreducible and positive recurrent, we can conclude that the Markov chain under study has a unique stationary distribution.

Proof. We will show that under the JRRS policy the Markov chain $(K^t(t))_{t=1,2,...}$ is irreducible and positive recurrent, we can conclude that the Markov chain under study has a unique stationary distribution.

First, we show that the Markov chain $(K^t(t))_{t=1,2,...}$ is irreducible. Specifically, observe that the Markov chain starting from any state $(k^t)_{t=1,2,...}, \xi^{1:L}(t), \eta^{1:L}(t)$ can reach the state $(k_0^{1:L}, (0)_{t=L+1,...}, 0^{1:L}, 0^{1:L})$, after experiencing a sequence of departures and arrivals that clears up all the tokens, virtual jobs and jobs on backup servers. Further, letting $\bar{k}$ be the configuration reachable by all other configuration in the single-server system under the policy $\bar{\sigma}$, we argue that starting from any states of the form $(k_0^{1:L}, (0)_{t=L+1,...}, 0^{1:L}, 0^{1:L})$, the Markov chain can reach the state $(\bar{k}^{1:L}, (0)_{t=L+1,...}, 0^{1:L}, 0^{1:L})$. Because for any $t \leq L$, there is a transition path from $k^t$ to $\bar{k}$, consider the sequence of events where each $K^t(t)$ transitions independently following the path, and the jobs arrive right after $K^t(t)$ making a request, so that the tokens are checked out before $K^t(t)$ has a further transition. In this way, each $K^t(t)$ with $t \leq L$ can eventually reach $\bar{k}$ from $k^t$. This proves the $k^0$-irreducibility of $(k^t(t))_{t=1,2,...}, \xi^{1:L}(t), \eta^{1:L}(t))$.

Next, we show that $(K^t(t))_{t=1,2,...}, \xi^{1:L}(t), \eta^{1:L}(t))$ satisfies Foster-Lyapunov criterion, i.e.

$$\tilde{G}g \leq -1 + b 1_{\{S\}},$$  

(37)
\[ + \sum_{i \in I} \left( \sum_{t=1}^{L_i} \sum_{(k',a) \in E(k')} \gamma_{k',(k',a)} a_i t_i - \sum_{t=1}^{\infty} \zeta_t \right) \] (41)

\[ \leq \sum_{i \in I} \left( \lambda_i t_i r + L \cdot \max_{k \in \mathcal{K}} \sum_{(k',a) \in E(k)} \gamma_{k,(k',a)} a_i t_i \right) \] (42)

\[ - \left( \sum_{t=1}^{\infty} \sum_{i \in I} k_t + \sum_{t=1}^{\infty} \sum_{i \in I} \zeta_t \right), \] (43)

where the first inequality uses the fact that the virtual jobs are generated at a rate no faster than the total rate of job requests. Then the Foster-Lyapunov criterion in (37) is satisfied with \( b \) and \( S \) given by

\[ b = \sum_{i \in I} \left( \lambda_i t_i r + L \cdot \max_{k \in \mathcal{K}} \sum_{(k',a) \in E(k)} \gamma_{k,(k',a)} a_i t_i \right), \]

\[ S = \left\{ ((K^\ell)_{\ell=1,2,\ldots}, \zeta_{1:1}, \eta_{1:1}): g((K^\ell)_{\ell=1,2,\ldots}, \zeta_{1:1}, \eta_{1:1}) \leq b + 1 \right\}. \]

By Foster-Lyapunov theorem, \( ((K^\ell(t))_{\ell=1,2,\ldots}, \zeta_{1:1}(t), \eta_{1:1}(t)) \) is positive recurrent. \qed

### A.2 Proof of Lemma 2

The proof of Lemma 2 follows the framework of Stein’s method (see, for example, [6–8]). To bound the distance between \( \overline{K}^{1:L} \) and \( \overline{R}^{1:L} \), observe that because \( f \in \operatorname{Lip}(1) \) and \( \sum_{\ell=1}^{L_i} \eta_{\ell}^\ell = O \left( r^{0.5} \right) \), it suffices to bound the Wasserstein distance between \( \overline{K}^{1:L} \) and \( \overline{R}^{1:L} + \eta^{1:L} \), i.e.,

\[ \sup_{f \in \operatorname{Lip}(1)} \left\{ \mathbb{E} \left[ f(\overline{K}^{1:L}) \right] - \mathbb{E} \left[ f(\overline{R}^{1:L} + \eta^{1:L}) \right] \right\} = O \left( r^{0.5} \right), \] (44)

where \( f(\overline{K}^{1:L} + \eta^{1:L}) \) is a valid expression because \( \overline{R}^{1:L} + \eta^{1:L} \in \mathcal{K} \) as discussed in (6.1). Stein’s method usually consists of three steps: generator comparison, Stein factor bounds, and moment bounds. In our case, due to the finiteness of the state space \( \mathcal{K} \), we only need to do the generator comparison and the Stein factor bounds. The generator comparison step justifies the intuition that most normal servers behave like single-server systems under the policy \( \overline{\sigma} \), and the Stein’s factor bounds corresponds to the intuition that the small fraction of servers whose dynamics deviate from single-server systems do not contribute much to the overall distance \( \mathbb{E} \left[ f(\overline{K}^{1:L}) \right] - \mathbb{E} \left[ f(\overline{R}^{1:L} + \eta^{1:L}) \right] \).

**System Dynamics and Generator.** To prepare for the proof, we first look into the dynamics of the two systems under study. In particular, we write out the *generators* of \( \overline{K}^{1:L}(t) \) and \( (\overline{K}^{1:L}(t), \eta^{1:L}(t)) \), which are used in the Stein’s method arguments.

We first examine the dynamics of the single-server system under Markovian policy \( \overline{\sigma} \). When the system has configuration \( \overline{k} \), the next transition should be either a departure or an internal transition potentially followed by a few new jobs being added to the service immediately, or a few jobs being added into service after an independent Poisson clock ticks. Both types of transitions can be represented by the following diagram:

\[ \overline{k} \rightarrow \overline{k}' \rightarrow \overline{k}' + \overline{a}, \]

which indicates that the system jumps from \( \overline{k} \) to \( \overline{k}' + \overline{a} \) for some \( \overline{k}' \in \mathcal{K} \) and \( \overline{a} = (a_i)_{i \in I} \) after an exponentially distributed amount of time, and \( \overline{a} \) jobs are requested during the jump. We denote the rate of the transition as \( \gamma_{\overline{k},(\overline{k}',\overline{a})} \), and the set of possible \( (\overline{k}',\overline{a}) \) pairs as \( E(\overline{k}) \).
We define the total transition rate at configuration $k$ as $\lambda_k = \sum_{(k',a) \in E(k)} \gamma_{k,k'}$, and define the maximal transition rate $\gamma_{\max} = \max_{k \in \mathcal{K}} \lambda_k$. Since $\mathcal{K}$ is a finite set, we must have $\gamma_{\max} < \infty$. Also observe that the request rate of type $i$ jobs is given by

$$\lambda_i = \sum_{k \in \mathcal{K}} \sum_{(k',a) \in E(k)} \gamma_{k,k'} a_i \cdot \pi(k),$$

where $\pi$ denotes the stationary distribution of single-server configuration under policy $\sigma$.

Next we focus on the dynamics of $L$ i.i.d. copies of single-server systems. Consider the generator $\overline{G}$ of the corresponding Markov chain $\{\overline{\mathcal{K}}(t)\}$, which is a linear operator on functions $g: \mathcal{K} \rightarrow \mathbb{R}$ defined as

$$\overline{G}g(k^{1:L}) \triangleq \frac{d}{dt} \mathbb{E} \left[ g\left(\overline{\mathcal{K}}^{1:L}(t)\right) \big| \overline{\mathcal{K}}^{1:L}(0) = k^{1:L} \right]_{t=0},$$

and we call the resulting function $\overline{G}g(\cdot)$ the drift of $g(\cdot)$. Based on the transition rates defined above, we have

$$\overline{G}g(k^{1:L}) = \sum_{\ell=1}^{L} \sum_{(k',a) \in E(k^\ell)} \gamma_{k^\ell,(k',a)} \left( g(\cdot, k^\ell + a, \cdot) - g(\cdot, k^\ell, \cdot) \right),$$

where $g(\cdot, k^\ell + a, \cdot) - g(\cdot, k^\ell, \cdot)$ is a shorthand for $g(k^1, \ldots, k^{\ell-1}, k^\ell + a, k^{\ell+1}, \ldots, k^L) - g(k^{1:L})$, i.e., we use $\cdot$ to omit the entries that agree with $k^{1:L}$.

Similarly, for the infinite-server system, consider the generator $\hat{G}$ of $\{\hat{\mathcal{K}}(t), \eta(t)\}$ defined as

$$\hat{G}g(k^{1:L}, \eta^{1:L}) \triangleq \frac{d}{dt} \mathbb{E} \left[ g\left(\hat{\mathcal{K}}^{1:L}(t), \eta^{1:L}(t)\right) \big| \hat{\mathcal{K}}^{1:L}(0) = k^{1:L}, \eta^{1:L}(0) = \eta^{1:L} \right]_{t=0},$$

for any function $g: (\mathcal{K} \times \mathcal{K})^{1:L} \rightarrow \mathbb{R}$. Observe that for each $\ell$, the transition of $(\hat{\mathcal{K}}^\ell(t), \eta^\ell(t))$ from $(k, \eta)$ to $(k^\ell, \eta + a \mathbb{1}_{\{\eta = 0\}})$ occurs at the rate $\gamma_{k^\ell,(k',a)}$ for each $(k',a) \in E(k)$, and any arrivals or virtual arrivals do not change the sum $\hat{\mathcal{K}}^\ell(t) + \eta^\ell(t)$. Therefore, if $\hat{G}$ is applied to a function $g(\cdot)$ that only depends on $k^{1:L} + \eta^{1:L}$, it has the form

$$\hat{G}g(k^{1:L}, \eta^{1:L}) = \sum_{\ell=1}^{L} \sum_{(k',a) \in E(k^\ell)} \gamma_{k^\ell,(k',a)} \left( g(\cdot, k^\ell + a, \cdot) - g(\cdot, k^\ell, \cdot) \right) \mathbb{1}_{\{\eta^\ell = 0\}}$$

$$+ \sum_{\ell=1}^{L} \sum_{(k',a) \in E(k^\ell)} \gamma_{k^\ell,(k',a)} \left( g(\cdot, k^\ell + \eta^\ell, \cdot) - g(\cdot, k^\ell, \cdot) \right) \mathbb{1}_{\{\eta^\ell \neq 0\}}.$$ (49)

Here we abuse the notation of $g$ to denote the function only depending on $k^{1:L} + \eta^{1:L}$. In this context $g(\cdot, k^\ell + \eta^\ell, \cdot) - g(\cdot, k^\ell, \cdot)$ is a shorthand for $g(k^1 + \eta^1, \ldots, k^{\ell-1} + \eta^{\ell-1}, k^\ell + \eta^\ell, k^{\ell+1} + \eta^{\ell+1}, \ldots, k^L + \eta^L) - g(k^{1:L} + \eta^{1:L})$, i.e., we use $\cdot$ to omit the entries that agree with $k^{1:L} + \eta^{1:L}$.

Now we are ready to present the proof.

**Proof of Lemma 2.** **Step 1: Generator Comparison.** For any $f \in \text{Lip}(1)$, consider the Poisson equation (see, for example, [6]) that solves for $g_f: \mathcal{K} \rightarrow \mathbb{R}$:

$$\mathbb{E} \left[ f(\hat{\mathcal{K}}^{1:L}) \right] - f(k^{1:L}) = \overline{G}g_f(k^{1:L}),$$

(50)

We let $k^{1:L} = \hat{\mathcal{K}}^{1:L} + \eta^{1:L}$ in (50) and take the expectation. This results in

$$\mathbb{E} \left[ f(\hat{\mathcal{K}}^{1:L}) \right] - \mathbb{E} \left[ f(\hat{\mathcal{K}}^{1:L} + \eta^{1:L}) \right] = \mathbb{E} \left[ \overline{G}g_f(\hat{\mathcal{K}}^{1:L} + \eta^{1:L}) \right].$$

(51)
On the other hand, because \((\hat{K}^{1:L}(t), \eta^{1:L}(t))\) is a finite-state Markov chain, we have
\[
E \left[ \hat{G} g_f (\hat{K}^{1:L} + \eta^{1:L}) \right] = 0. \tag{52}
\]
Subtracting (52) from (51), we get
\[
E \left[ f (\hat{K}^{1:L}) \right] - E \left[ f (\hat{K}^{1:L} + \eta^{1:L}) \right] = E \left[ (\hat{G} - \hat{G}) g_f (\hat{K}^{1:L} + \eta^{1:L}) \right]. \tag{53}
\]
We want to show that \(\hat{G} - \hat{G}\) are close so that we can bound the RHS of (53).

Now we plug in the formula of the generators in (47) and (49) to the RHS of (53) and get
\[
\left| (\hat{G} - \hat{G}) g_f (\hat{K}^{1:L} + \eta^{1:L}) \right| = \left| \sum_{\ell=1}^{L} \sum_{(k',a) \in E(\hat{K}^{\ell} + \eta^{\ell})} \gamma_{\hat{K}^{\ell}+\eta^{\ell},(k',a)} \left( g_f (\cdot, k' + a, \cdot) - g_f (\cdot, \hat{K}^{\ell} + \eta^{\ell}, \cdot) \right) \cdot 1_{\eta^{\ell} \neq 0} - \sum_{\ell=1}^{L} \sum_{(k',a) \in E(\hat{K}^{\ell})} \gamma_{\hat{K}^{\ell},(k',a)} \left( g_f (\cdot, k' + \eta^{\ell}, \cdot) - g_f (\cdot, \hat{K}^{\ell} + \eta^{\ell}, \cdot) \right) \cdot 1_{\eta^{\ell} \neq 0} \right| \tag{54}
\]
\[
\leq \sum_{\ell=1}^{L} \gamma_{\hat{K}^{\ell}+\eta^{\ell}} \cdot \sup_{(k',a) \in E(\hat{K}^{\ell} + \eta^{\ell})} \left| g_f (\cdot, k' + a, \cdot) - g_f (\cdot, \hat{K}^{\ell} + \eta^{\ell}, \cdot) \right| \cdot 1_{\eta^{\ell} \neq 0} + \sum_{\ell=1}^{L} \gamma_{\hat{K}^{\ell}} \cdot \sup_{(k',a) \in E(\hat{K}^{\ell})} \left| g_f (\cdot, k' + \eta^{\ell}, \cdot) - g_f (\cdot, \hat{K}^{\ell} + \eta^{\ell}, \cdot) \right| \cdot 1_{\eta^{\ell} \neq 0} \tag{55}
\]
\[
\leq 2\gamma_{\max} \cdot \sum_{\ell=1}^{L} \sup_{k',a \in K} \left| g_f (\cdot, k', \cdot) - g_f (\cdot, \hat{K}^{\ell} + \eta^{\ell}, \cdot) \right| \cdot 1_{\eta^{\ell} \neq 0} + \sum_{\ell=1}^{L} \sum_{\ell \in I} 1_{\eta^{\ell} \neq 0}. \tag{56}
\]
where in \(g_f (\cdot, k', \cdot) - g_f (\cdot, k, \cdot)\) we have omit the entries that agree with \(k^{1:L}\). The equality is true because each of the \(\ell\)-th terms in \(\overline{G}\) and \(\hat{G}\) is equal if \(\eta^{\ell} = 0\) for all \(i\). For the first and second inequalities, recall that \(\gamma_k\) is the total transition rate given by \(\gamma_k = \sum_{(k',a) \in E(k)} \gamma_k(k',a).\) and \(\gamma_{\max} = \max_{k \in K} \gamma_k\). Observe that
\[
\sum_{\ell=1}^{L} 1_{\eta^{\ell} \neq 0} \leq \sum_{\ell=1}^{L} \sum_{i \in I} \eta^{\ell}_i \leq |I| \cdot \eta_{\max} = O (r^{0.5}).
\]
therefore (57) can be further bounded by
\[
\left| (\overline{G} - \hat{G}) g_f (\hat{K}^{1:L} + \eta^{1:L}) \right| \leq 2\gamma_{\max} \cdot \sup_{k,k' \in K} \left| g_f (\cdot, k', \cdot) - g_f (\cdot, k, \cdot) \right| \cdot \sum_{\ell=1}^{L} \sum_{i \in I} \eta^{\ell}_i \leq 2\gamma_{\max} \cdot \sup_{k,k' \in K} \left| g_f (\cdot, k', \cdot) - g_f (\cdot, k, \cdot) \right| \cdot O (r^{0.5}). \tag{58}
\]
To prove (44), it remains to show that
\[
\sup_{k,k' \in K} \left| g_f (\cdot, k', \cdot) - g_f (\cdot, k, \cdot) \right| = O (1). \tag{58}
\]
**Step 2: Stein Factor Bound.** To prove (58), observe that the following \( g_f(\cdot) \) is a solution to the Poisson equation (50). (see, for example, [6])

\[
g_f(k^{1:L}) = \mathbb{E} \left[ \int_0^\infty \left( f(\overline{K}^{1:L}(t)) - \mathbb{E} \left[ f(\overline{K}^{1:L}(t)) \right] \right) dt \right| \overline{K}^{1:L}(0) = k^{1:L} \right]. \tag{59}
\]

This allows us to bound the difference of \( g_f \) using coupling. Specifically, we define the coupling of two systems, each consisting of \( L \) i.i.d. copies of the single-server system under \( \overline{\sigma} \). The two systems are initialized with configurations \( (\cdot, k', \cdot) \) and \( (\cdot, k, \cdot) \) that only differ at the \( \ell \)-th server, where we omit the entries that agree with \( k^{1:L} \). Let \( (\overline{K}^{1:L,1}(t), \overline{K}^{1:L,2}(t)) \) be the joint configuration of the two systems, which is actually \( 2L \) i.i.d. copies of the single-server system. As a result, we can specify the couplings \( (\overline{K}^{1:L,1}(t), \overline{K}^{1:L,2}(t)) \) for different \( t' \) separately. For \( t' \neq t \), the corresponding server in the two systems have the same initial configurations, so we can always keep their configurations identical. For the \( \ell \)-th servers, we let them evolve independently following their own dynamics until a stopping time \( \tau_{mix} \), when their configurations meet. After that, we can use coupling to keep their configurations identical. Under this coupling, it is not hard to see that

\[
|g_f(\cdot, k', \cdot) - g_f(\cdot, k, \cdot)| = \mathbb{E} \left[ \int_0^\infty \left( f(\overline{K}^{1:L,1}(t)) - f(\overline{K}^{1:L,2}(t)) \right) dt \right] \\
\leq \mathbb{E} \left[ \int_0^\infty \| f(\overline{K}^{1:L,1}(t)) - f(\overline{K}^{1:L,2}(t)) \| dt \right] \\
\leq \mathbb{E} \left[ \int_0^\infty \| \overline{K}^{1:L,1}(t) - \overline{K}^{1:L,2}(t) \| dt \right] \\
= \mathbb{E} \left[ \int_0^\infty \sum_{\ell=1}^L \| \overline{K}^{1:L,1}(t) - \overline{K}^{1:L,2}(t) \| dt \right] \\
= \mathbb{E} \left[ \int_0^{\tau_{mix}} \| \overline{K}^{1:L,1}(t) - \overline{K}^{1:L,2}(t) \| dt \right]. \tag{60}
\]

For each pair of \( k, k' \), observe that because \( \overline{\sigma} \) is a \( k^0 \)-irreducible policy, \( \mathbb{E}[\tau_{mix}] \) is finite; and because \( \mathcal{K} \) is a finite set, \( \| \overline{K}^{1:L,1}(t) - \overline{K}^{1:L,2}(t) \| \) is uniformly bounded. All these finite quantities depend on a single-server system under a policy \( \overline{\sigma} \) that is independent of \( r \). As a result, the last expression in (60) is of constant order. Moreover, because there are finite pairs of \( (k, k') \), the supremum \( \sup_{k, k'} \mathbb{E} \left[ \int_0^{\tau_{max}} \| \overline{K}^{1:L,1}(t) - \overline{K}^{1:L,2}(t) \| dt \right] \) is also of constant order, independent of \( r \). This proves the Stein factor bound in (58). Together with the generator comparison, we have proved

\[
\sup_{f \in \text{Lip}(1)} \mathbb{E} \left[ f(\overline{K}^{1:L}) \right] - \mathbb{E} \left[ f(\overline{K}^{1:L} + \eta^{1:L}) \right] = O \left( r^{0.5} \right). \tag{44}
\]

The goal of the lemma is a straightforward consequence of (44), as shown below. Because \( \sum_{t=1}^L \sum_{i \in \mathcal{I}} \eta_i^t \leq |\mathcal{I}| \eta_{max} = O(r^{0.5}) \), for any \( f \in \text{Lip}(1) \), we have

\[
\mathbb{E} \left[ f(\overline{K}^{1:L}) \right] - \mathbb{E} \left[ f(\overline{K}^{1:L} + \eta^{1:L}) \right] \leq \mathbb{E} \left[ \sum_{t=1}^L \sum_{i \in \mathcal{I}} \eta_i^t \right] = O \left( r^{0.5} \right). \tag{61}
\]

Therefore,

\[
\sup_{f \in \text{Lip}(1)} \mathbb{E} \left[ f(\overline{K}^{1:L}) \right] - \mathbb{E} \left[ f(\overline{K}^{1:L}) \right] = O \left( r^{0.5} \right). \tag{62}
\]

This finishes the proof.
A.3 Proof of Lemma 3 and Lemma 4

In this subsection, we prove Lemma 3 and Lemma 4 together. We begin by introducing some notations. We use $U^f \doteq (\tilde{K}^f, \eta^f)$ to represent the state of the $f$-th server, and use $U^{1:L}$ to represent the joint state of the first $L$ servers. We also use the lower case $u^f, u^{1:L}$ to represent the realizations of the corresponding random variables. We denote the total number of type $i$ virtual jobs as $V_i \doteq \sum_{f=1}^{L} \zeta^f_i$ for $i \in I$, and its realization as $v_i$. We denote the total number of type $i$ jobs on backup servers as $Y_i$ for $i \in I$, and its realizations as $y_i$. We also denote the total number of type $i$ tokens throughout the system as $Z_i \doteq \sum_{f=1}^{L} \eta^f_i$, and its realization as $z_i$. Our goal can be rewritten as proving $\mathbb{E}[V_i] = O (\rho^{0.5})$ and $\mathbb{E}[Y_i] = O (r^{0.5})$ for each $i \in I$.

We first give an overview of the proof. Observe that in our model, the expected time that a jobs stay in the system is fixed. As a result, bounding the number of virtual jobs or jobs on backup servers in the system is equivalent to bounding the rate that they are generated, according to Little’s Law. By our construction of the policy, the rate of generating those jobs are closely related to the dynamics of the total number of type $i$ tokens $Z_i(t)$.

To describe the dynamics of $Z_i(t)$, we first introduce two functions $dv_i$ and $dy_i$:

$$
dv_i(a_i, z_i) \doteq (z_i + a_i - \eta_{\text{max}})^+,
$$

$$
dy_i(z_i) \doteq (1 - z_i)^+.
$$

The function $dv_i$ represents the increment in the number of type $i$ virtual jobs due to the event that the total number of type $i$ tokens on the first $L$ servers exceeds the token limit $\eta_{\text{max}}$. The function $dy_i$ corresponds to the increment in the total number of type $i$ jobs on backup servers due to the event that a type $i$ job arrives to the system without seeing a type $i$ token. For a function $g$: $(\mathcal{K} \times \mathcal{K})^L \to \mathbb{R}$ that only depends on the number of type $i$ tokens $z_i$, its drift can be written as

$$
\tilde{G}g(u^{1:L}) = \sum_{f=1}^{L} \sum_{(k', a) \in E(k^f)} Y_{k^f,(k', a)} (g(z_i + a_i - \tilde{d}v_i(a_i, z_i)) - g(z_i)) \mathbbm{1}_{(\eta' = 0)} + \lambda_i r (g(z_i - 1 + \tilde{d}y_i(z_i)) - g(z_i)).
$$

We abuse the notation of $g$ here. For ease of exposition, we will simply write $dv_i$ and $dy_i$ to represent $dv_i(a_i, z_i)$ and $dy_i(z_i)$.

By construction, the total number of type $i$ tokens $\{Z_i(t)\}$ is a stochastic process constrained within $[0, \eta_{\text{max}}]$. Note that $Z_i(t)$ increases when some servers request new tokens, and decreases when a real or virtual arrival checks out the token or when some servers have the excessive tokens removed. When $Z_i(t)$ is away from the boundaries, the average rate that it increases is approximately equal to $\lambda_i r$, and the average rate that $Z_i(t)$ decreases is given by

$$
\mathbb{E} \left[ \sum_{f=1}^{L} \sum_{(k', a) \in E(\tilde{K}^f)} Y_{\tilde{K}^f,(k', a)} a_i \mathbbm{1}_{(\eta' = 0)} \right] \approx \mathbb{E} \left[ \sum_{f=1}^{L} \sum_{(k', a) \in E(\tilde{K}^f)} Y_{\tilde{K}^f,(k', a)} a_i \right] = L \cdot \tilde{\lambda}_i \approx \lambda_i r,
$$

where we have used the approximations that $\tilde{K}^f \doteq \tilde{K}^f, \mathbbm{1}_{(\eta' = 0)} \approx 1$ and $L = \lceil N \rceil \approx N$.

As $\{Z_i(t)\}$ randomly moves up and down with approximately the same rate and reflects on the boundaries of 0 and $\eta_{\text{max}}$, it behaves as a reflected simple symmetric random walk. Intuitively speaking, the steady-state distribution of $Z_i$ is approximately an uniform distribution over $[0, \eta_{\text{max}}]$. Recall that $dv_i$ and $dy_i$ can only be non-zero when $Z_i(t)$ is near the boundaries. Since the length of the interval $\eta_{\text{max}} = \Theta (\rho^{0.5})$, we can expect that $dv_i$ and $dy_i$ diminish as $r \to \infty$.

In the proof, we first establish the relationship between $\mathbb{E}[V_i], \mathbb{E}[Y_i]$ and $dv_i, dy_i$ using Little’s Law. Then we derive bounds on $dv_i$ and $dy_i$ by analyzing the drift of several test functions of $Z_i$. This
We establish the relationships of \( Z_i \) with the tokens being generated and eliminated at similar rates. Finally, we invoke Lemma 2 to show that the tokens are indeed generated and eliminated at similar speeds, which leads to bounds on \( dv_i \) and \( dy_i \).

Finally, we make some additional remarks on the notations. First, \( dv_i \) and \( dy_i \) depend on total number of type \( i \) tokens \( z_i \) and the number of newly requested jobs \( a_i \), although we omit the dependency expression for ease of exposition. Second, we abuse the notation \( dv_i \) and \( dy_i \) to denote the corresponding random variables. We also write \( \sum_{(k',a')} \) as a shorthand for \( \sum_{(k',a') \in E(k')} \) when the context is clear.

**Proof of Lemma 3 and Lemma 4. Bounding Virtual Jobs and Jobs on Backup Servers using Little’s Law.** We first apply Little’s Law to \( V_i \) and \( Y_i \). Because the expected time that a job stays in the system is no more than \( \frac{1}{\mu_{\text{min}}} \) with \( \mu_{\text{min}} \equiv \min_i \mu_i \), we have

\[
\mathbb{E}[V_i] \leq \frac{1}{\mu_{\text{min}}} \mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} dv_i 1_{\{\eta' = 0\}} \right],
\]

\[
\mathbb{E}[Y_i] \leq \frac{1}{\mu_{\text{min}}} \mathbb{E} [\lambda_i r \cdot dy_i].
\]

**Drift Analysis.** The above two equations (64) and (65) suggest that we can derive upper bounds on \( \mathbb{E}[V_i] \) and \( \mathbb{E}[Y_i] \) by analyzing the following two terms:

- \( \mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} dv_i 1_{\{\eta' = 0\}} \right] \), interpreted as the average rate that \( Z_i(t) \) reflects on the boundary at \( \eta_{\text{max}} \);
- \( \mathbb{E} [\lambda_i r \cdot dy_i] \), interpreted as the average rate that \( Z_i(t) \) reflects on the boundary at 0.

We establish the relationships of \( dv_i \), \( dy_i \) and \( Z_i \) by analyzing the drift of two test functions \( g \).

Letting \( g(z_i) = z_i \) and taking steady-state expectation over its drift, by (63) and the fact that the drift is zero in steady state, we get

\[
\mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} (a_i - dv_i) 1_{\{\eta' = 0\}} + \lambda_i r (-1 + dy_i) \right] = 0.
\]

Similarly, letting \( g(z_i) = z_i^2 \) and taking steady-state expectation over its drift, one can verify that

\[
\mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} dv_i 1_{\{\eta' = 0\}} \right] = \frac{1}{\eta_{\text{max}}} \mathbb{E} \left[ \left( \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} a_i 1_{\{\eta' = 0\}} - \lambda_i r \right) \cdot Z_i \right] + \frac{1}{2\eta_{\text{max}}} \mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} (a_i^2 - (dv_i)^2) 1_{\{\eta' = 0\}} + \lambda_i r \cdot (1 - (dy_i)^2) \right].
\]

Readers may refer to the complete calculation at the end of this subsection.

**Estimating the Terms Obtained from Drift Analysis.** We will first focus on bounding \( \mathbb{E} \left[ \sum_{\ell=1}^L \sum_{(k,a')} Y_{K',(k',a')} dv_i 1_{\{\eta' = 0\}} \right] \) analyzing the two terms in (68) and (69) separately. Then we invoke (6) to bound \( \mathbb{E} [\lambda_i r \cdot dy_i] \).
The term in (69) is easy to deal with. Observe that the number of jobs requested each time should be no more than the maximal number of jobs that a server can hold, i.e. \( a_i \leq K_{\text{max}} \), so

\[
(69) \leq \frac{1}{2r_{\text{max}}} \cdot \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} K_{\text{max}}^2 + \lambda_i r \right]
\]

\[
\leq \frac{1}{2r_{\text{max}}} \cdot \mathbb{E} \left[ \sum_{\ell=1}^{L} Y_{\text{max}} K_{\text{max}}^2 + \lambda_i r \right]
\]

\[
= O \left( r^{0.5} \right),
\]

where in the second inequality we have used the fact that the total rate is uniformly bounded by \( y_{\text{max}} \), and the last step uses the facts that \( L = O \left( r \right) \) and \( r_{\text{max}} = \Theta \left( r^{0.5} \right) \).

To bound the term in (68), first observe that \( Z_i \leq \eta_{\text{max}} \), which implies that

\[
(68) \leq \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i I_{\{\eta'=0\}} - \lambda_i r \right].
\]

The term on the RHS of the above equation is the expected absolute difference between the rates of generating and eliminating type \( i \) tokens, which is can be shown to be small relative to \( r \). Specifically, we claim that

\[
\mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i I_{\{\eta'=0\}} - \lambda_i r \right] = O \left( r^{0.5} \right).
\]

To show (71), first notice that we can remove the indicator \( 1_{\{\eta'=0\}} \) without introducing much error:

\[
\mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i I_{\{\eta'=0\}} - \lambda_i r \right] \leq \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i - \lambda_i r \right] + \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i I_{\{\eta'\neq0\}} \right]
\]

\[
\leq \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i - \lambda_i r \right] + \sum_{\ell=1}^{L} Y_{\text{max}} K_{\text{max}} I_{\{|I| \leq \eta_{\text{max}}\}}
\]

where the first inequality is due to triangular inequality, the second inequality is due to the definition of \( y_{\text{max}} \), and the last inequality is due to the fact that \( \sum_{\ell=1}^{L} I_{\{\eta'\neq0\}} \leq |I| \eta_{\text{max}} \). It remains to bound the term \( \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i - \lambda_i r \right] \), which can be seen as showing that the rate of generating type \( i \) tokens concentrates around the type \( i \) jobs’ arrival rate \( \lambda_i r \). It is natural to think of using some Law of Large Numbers. Unfortunately, \( \sum_{\ell=1}^{L} \sum_{(k,a')} Y_{\tilde{K}^\ell,(k',a')} a_i \) is not a sum of i.i.d. random variables due to dependencies among \( \tilde{K}^\ell \) for different \( \ell \)’s. As a result, we want to invoke the Wasserstein distance bound in Lemma 2 to replace \( \tilde{K}^\ell \) in the above expression with \( K_{\text{\ell}} \). We define
the function \( f(k^{1:L}) \) as
\[
f(k^{1:L}) = \frac{1}{2\gamma_{\text{max}}K_{\text{max}}} \left| \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right| .
\]

We claim that \( f \in \text{Lip}(1) \). For any two \( k^{1:L,1}, k^{1:L,2} \),
\[
2\gamma_{\text{max}}K_{\text{max}} \cdot \left| f(k^{1:L,1}) - f(k^{1:L,2}) \right|
\leq \sum_{\ell=1}^{L} \left| \sum_{(k,a')} \gamma_{k^{\ell,1},(k',a')} a_i - \lambda_i r \right| - \sum_{\ell=1}^{L} \left| \sum_{(k,a')} \gamma_{k^{\ell,2},(k',a')} a_i - \lambda_i r \right|
\leq \sum_{\ell=1}^{L} \sum_{(k,a')} \left| \gamma_{k^{\ell,1},(k',a')} a_i - \lambda_i r \right| - \sum_{\ell=1}^{L} \sum_{(k,a')} \left| \gamma_{k^{\ell,2},(k',a')} a_i - \lambda_i r \right|
\leq \sum_{\ell=1}^{L} \sum_{(k,a')} \left| \gamma_{k^{\ell,1},(k',a')} - \gamma_{k^{\ell,2},(k',a')} \right| a_i \mathbb{1}_{\{k^{\ell,1} \neq k^{\ell,2}\}}
\leq \sum_{\ell=1}^{L} \left| k^{\ell,1} - k^{\ell,2} \right|
= 2\gamma_{\text{max}}K_{\text{max}} \cdot \left| k^{1:L,1} - k^{1:L,2} \right|,
\]
where the first inequality is due to triangular inequality; the second inequality uses the fact that \( a_i \leq K_{\text{max}} \) and \( \mathbb{1}_{\{k^{\ell,1} \neq k^{\ell,2}\}} \leq \left| k^{\ell,1} - k^{\ell,2} \right|; \) the third inequality uses triangular inequality, the fact that the total rate at a configuration \( k \) is bounded by \( \gamma_{\text{max}} \) and the property of the \( L^1 \) norm \( \| \cdot \| \). Therefore, \( f \in \text{Lip}(1) \). The Lipschitz continuity of \( f \) allows us to invoke Lemma 2 and get
\[
\mathbb{E} \left[ \left| \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right| - \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right] \right] \leq 2\gamma_{\text{max}}K_{\text{max}} \cdot O \left( r^{0.5} \right).
\]

Therefore,
\[
\mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right] \leq \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right] + O \left( r^{0.5} \right).
\]

Observe that under a Markovian policy, the request rate of type \( i \) jobs \( \tilde{\lambda}_i = \mathbb{E}[\sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i] \), so we have
\[
\mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right] = \tilde{\lambda}_i \cdot \lceil N \rceil - \lambda_i r = O \left( 1 \right).
\]

Moreover, because \( \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i \) are i.i.d. for \( \ell = 1, \ldots, L \), we have
\[
\mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right] \leq \mathbb{E} \left[ \left( \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k^{\ell},(k',a')} a_i - \lambda_i r \right)^2 \right] + O \left( 1 \right).
\]
which proves (71). This implies that the term in (68) is also in $O\left(r^{0.5}\right)$. Combining the bounds on the terms in (68) and (69), we get

$$
\mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i \mathbb{I}_{\{\eta' = 0\}} - \lambda_i r \right] \leq \mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i - \lambda_i r \right] + O\left(r^{0.5}\right)
$$

$$
\leq \mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i - \lambda_i r \right] + O\left(r^{0.5}\right)
$$

$$
\leq O\left(r^{0.5}\right),
$$

which proves (71). This implies that the term in (68) is also in $O\left(r^{0.5}\right)$. Combining the bounds on the terms in (68) and (69), we get

$$
\mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} \tilde{d}_{q_l} \mathbb{I}_{\{\eta' = 0\}} \right] = O\left(r^{0.5}\right).
$$

Finally, we bound $\mathbb{E} \left[ \lambda_i r \cdot d y_i \right]$. We rearrange the terms in (66) and get

$$
\mathbb{E} \left[ \lambda_i r \cdot d y_i \right] = \mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} (-a_i + d v_l) \mathbb{I}_{\{\eta' = 0\}} + \lambda_i r \right]
$$

$$
= \mathbb{E} \left[ - \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i \mathbb{I}_{\{\eta' = 0\}} + \lambda_i r \right] + O\left(r^{0.5}\right).
$$

By (71), we have $\mathbb{E} \left[ - \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i \mathbb{I}_{\{\eta' = 0\}} + \lambda_i r \right] = O\left(r^{0.5}\right)$. Therefore,

$$
\mathbb{E} \left[ \lambda_i r \cdot d y_i \right] = O\left(r^{0.5}\right).
$$

We invoke the equations (64) and (65) that we get at the beginning of the proof, and conclude that

$$
\mathbb{E}[V_i] \leq \frac{1}{\mu_{\text{min}}} \mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} \tilde{d}_{q_l} \mathbb{I}_{\{\eta' = 0\}} \right] = O\left(r^{0.5}\right).
$$

$$
\mathbb{E}[Y_i] \leq \frac{1}{\mu_{\text{min}}} \mathbb{E} \left[ \lambda_i r \cdot d y_i \right] = O\left(r^{0.5}\right).
$$

This finishes the proof. \qed

**Deriving the equality in (67).** We show the calculation detail of deriving the following equality.

$$
\mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} \tilde{d}_{q_l} \mathbb{I}_{\{\eta' = 0\}} \right] = O\left(r^{0.5}\right).
$$

$$
\mathbb{E} \left[ \sum_{l=1}^{L} \sum_{(k,a')} Y_{\hat{l},(k',a')} a_i \mathbb{I}_{\{\eta' = 0\}} \right] - \lambda_i r \cdot Z_i
$$

(68)
\[ + \frac{1}{2\eta_{\text{max}}} \cdot \mathbb{E} \left[ \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k',(k',a')}^{\ell} \left( a_i^2 - (d\eta_i)^2 \right)^2 \right] \cdot 1_{\{\eta'_{\ell}=0\}} + \lambda_i r \cdot (1 - (dy_i)^2) - (75). \] (69)

The equality is obtained by considering the drift of the function \( g(z_i) = z_i^2 \), which is zero in steady state. Recall that the drift of \( g(z_i) \) is given by

\[
\hat{G} g(z_i, u^1:L) = \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k',(k',a')}^{\ell} \left( g(z_i + a_i - d\eta_i) - g(z_i) \right) \cdot 1_{\{\eta'_{\ell}=0\}} + \lambda_i r \cdot (g(z_i - 1 + dy_i) - g(z_i)). \] (63)

We will first calculate \( g(z_i + a_i - d\eta_i) - g(z_i) \), then \( g(z_i - 1 + dy_i) - g(z_i) \), and finally plug them into the (63).

The calculation of \( g(z_i + a_i - d\eta_i) - g(z_i) \) utilizes the following property of \( d\eta_i \):

\[
(z_i + a_i - d\eta_i) \cdot d\eta_i = \eta_{\text{max}} \cdot d\eta_i. \] (75)

This property follows from the definition \( d\eta_i = (z_i + a_i - \eta_{\text{max}})^+ \). Intuitively, this is because \( d\eta_i \) is the "force" that pushes \( z_i \) back when it hits the boundary at \( \eta_{\text{max}} \). Using the property, we have

\[
g(z_i + a_i - d\eta_i) - g(z_i) = (z_i + a_i - d\eta_i)^2 - z_i^2 = (z_i + a_i - d\eta_i)^2 - (z_i + a_i - d\eta_i - a_i + d\eta_i)^2 \]

\[= (z_i + a_i - d\eta_i)^2 - ((z_i + a_i - d\eta_i)^2 + 2(-a_i + d\eta_i) \cdot (z_i + a_i - d\eta_i) + (-a_i + d\eta_i)^2) \]

\[= 2(a_i - d\eta_i) \cdot (z_i + a_i - d\eta_i) - (-a_i + d\eta_i)^2 \]

\[= 2a_i \cdot (z_i + a_i - d\eta_i) - (-a_i + d\eta_i)^2 - 2d\eta_i \cdot \eta_{\text{max}} \]

\[= 2a_i \cdot z_i + a_i^2 - (d\eta_i)^2 - 2d\eta_i \cdot \eta_{\text{max}}. \]

The second last equality is due to (75), and the rest are all algebraic manipulations.

We carry out a similar calculation for \( g(z_i - 1 + d\eta_i) - g(z_i) \):

\[
g(z_i - 1 + d\eta_i) - g(z_i) = 2(-1 + d\eta_i) \cdot z_i + (-1 + d\eta_i)^2 \]

\[= -2z_i + 2z_i \cdot d\eta_i + 1 - 2d\eta_i + (d\eta_i)^2 \]

\[= -2z_i + 1 + 2(z_i - 1 + d\eta_i) \cdot d\eta_i - (d\eta_i)^2 \]

\[= -2z_i + 1 - (d\eta_i)^2. \]

where the last equality is due to the property that

\[
(z_i - 1 + d\eta_i) \cdot dy_i = 0, \] (76)

and the rest are all algebraic manipulations.

Putting together,

\[
\hat{G} g(z_i, u^1:L) = \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k',(k',a')}^{\ell} \left( g(z_i + a_i - d\eta_i) - g(z_i) \right) \cdot 1_{\{\eta'_{\ell}=0\}} + \lambda_i r \cdot (g(z_i - 1 + dy_i) - g(z_i)) \]

\[= \sum_{\ell=1}^{L} \sum_{(k,a')} \gamma_{k',(k',a')}^{\ell} \left( 2a_i \cdot z_i + a_i^2 - (d\eta_i)^2 - 2d\eta_i \cdot \eta_{\text{max}} \right) \cdot 1_{\{\eta'_{\ell}=0\}} \]
After recombining the terms, we get

\[ + \lambda_i r \cdot (-2z_i + 1 - (dy_i)^2). \]

After recombining the terms, we get

\[
\eta_{\text{max}} \cdot \mathbb{E} \left[ \sum_{t=1}^{L} \sum_{(k,a')} Y_{\bar{K}_t,(k',a')} d\eta_t \mathbb{1}_{\eta_t = 0} \right] \\
= \mathbb{E} \left[ \left( \sum_{t=1}^{L} \sum_{(k,a')} Y_{\bar{K}_t,(k',a')} a_t \mathbb{1}_{\eta_t = 0} - \lambda_i r \right) \cdot Z_t \right] \\
+ \frac{1}{2} \mathbb{E} \left[ \sum_{t=1}^{L} \sum_{(k,a')} Y_{\bar{K}_t,(k',a')} (a_t^2 - (d\eta_t)^2) \mathbb{1}_{\eta_t = 0} + \lambda_i r \cdot (1 - (dy_i)^2) \right].
\]

This finishes the calculation.

### A.4 Proof of Lemma 5

**Proof.** We prove by contradiction. Suppose that ML(S) \( \not\subseteq S \), then both ML(S) \( \cap S \) and ML(S) \( \setminus S \) are non-empty. Let \( k \in \text{ML}(S) \cap S \) and \( k' \in \text{ML}(S) \setminus S \). Observe that because the jobs’ phases follow an irreducible Markov chain and \( \sum_i k_i = \sum_i k'_i \), if we only consider internal transitions, \( k \) can become \( k' \) through a finite number of internal transitions. Without loss of generality, we assume that \( k \) can become \( k' \) through one internal transition. Because \( k' \notin S \), the policy \( \bar{\sigma} \) must request some jobs immediately when the system reaches \( k' \) from \( k \). However, this means \( k \) can reach another configuration \( k'' \) such that \( \sum_i k''_i > \sum_i k'_i = \sum_i k_i \). Consequently, \( k'' \in S \), contradicting the fact that \( k \in \text{ML}(S) \).

\( \square \)

### A.5 Proof of Lemma 6

**Proof.** For each \( j = 1, 2, \ldots, J \), we define the policy \( \bar{\sigma}^j \) as follows: when the system has configuration \( k \in S_j \), the policy \( \bar{\sigma}^j \) makes the same decisions as \( \bar{\sigma} \). When \( k \notin S_j \), the policy \( \bar{\sigma}^j \) requests jobs in the following cases:

- When the system has configuration \( k = 0 \), the policy runs a Poisson clock with rate 1 for each type \( i \) job and requests a job of the corresponding type when the clock ticks;
- After a new job has been requested or after a departure event or internal transition event, if the current configuration \( k \) satisfies \( \sum_i k_i < \max_{k' \in S_j} \sum_i k'_i \), the policy immediately requests a job of random type.

We show that under the new policy \( \bar{\sigma}^j \), \( S_j \) is also a recurrent class of the induced Markov chain. This is because if the system starts from a configuration \( k \in S_j \), then it will stay in \( S_j \) since it makes the same decisions as \( \bar{\sigma} \), and the transitions will also be the same as under the policy \( \bar{\sigma} \). Because \( S_j \) is a recurrent class under \( \bar{\sigma} \), it is still a recurrent class under \( \bar{\sigma}^j \).

To show that the Markov chain induced by \( \bar{\sigma}^j \) is \( k^0 \)-irreducible, observe that starting from any \( k \notin S_j \), the system state will return to \( S_j \). Specifically,

- If the system starts from a configuration \( k \) such that \( k \notin S_j \) and \( \sum_i k_i > \max_{k' \in S_j} \sum_i k'_i \), then no new jobs will be requested until \( k \in S_j \) or \( \sum_i k_i = \max_{k' \in S_j} \sum_i k'_i \), and in the latter case we also have \( k \in S_j \) according to Lemma 5;
- If the system starts from a configuration \( k \) such that \( k \notin S_j \) and \( \sum_i k_i < \max_{k' \in S_j} \sum_i k'_i \), when the next event happens, the policy immediately requests a series of jobs until \( k \in S_j \) or \( \sum_i k_i = \max_{k' \in S_j} \sum_i k'_i \), and in the latter case we also have \( k \in S_j \).
The fact that the stationary distribution under $\tilde{\sigma}^f$ is the same as the stationary distribution under $\tilde{\sigma}$ starting from a configuration in $S_f$ trivially follows from the fact that the sample paths under the two policies can be exactly coupled. \qed

B  MISSING PROOFS FOR SINGLE-SERVER SYSTEM

B.1 Proof of Lemma 8

Proof. Let $(\pi, (u_i)_{i \in I})$ be a feasible solution to the LP in (10). To show that $\pi$ and $(u_i)_{i \in I}$ are also the actual stationary distribution and transition frequencies, it suffices to show that if the initial distribution follows $\pi$, i.e.

$$\Pr(\tilde{K}(0) = k) = \pi(k)$$

then under the policy $\tilde{\sigma}(\pi, (u_i)_{i \in I})$, we have

$$\lim_{T \to \infty} \frac{1}{T} \int_0^T \Pr(\tilde{K}(t) = k) \, dt = \pi(k),$$

where $F$ is the cumulative number of nominal transitions under the policy $\tilde{\sigma}(\pi, (u_i)_{i \in I})$ and the initial distribution $\pi$.

Our proof is based on the following equation:

$$\frac{d}{dt} \Pr(\tilde{K}(t) = k)\bigg|_{t=0} = \sum_i \frac{d}{dt} \mathbb{E}[F(k - e_i, k, t)]\bigg|_{t=0} - \sum_i \frac{d}{dt} \mathbb{E}[F(k, k + e_i, t)]\bigg|_{t=0} + \sum_i (k_i + 1) \mu_i \pi(k + e_i) + \sum_{i,i'} (k_i + 1) \mu_{i,i'} \pi(k + e_i - e_i') \mathbbm{1}_{\{k_i \geq 1\}} - \sum_i k_i \mu_i \pi(k) - \sum_{i,i'} k_i \mu_{i,i'} \pi(k),$$

(79)

The equation is a straightforward consequence of (33), following the same argument as the proof of Lemma 7.

We prove the following two equations by induction on $\sum_{i \in I} k_i$.

$$\lim_{t \to 0} \frac{1}{t} \mathbb{E}[F(k, k + e_i, t)] = u_i(k),$$

(80)

$$\frac{d}{dt} \Pr(\tilde{K}(t) = k)|_{t=0} = 0.$$  

(81)

We first consider the base case when $\sum_{i \in I} k_i = 0$. In this case, $k = 0$ and we have

$$\frac{d}{dt} \mathbb{E}[F(k - e_i, k, t)]\bigg|_{t=0} = u_i(k - e_i) \mathbbm{1}_{\{k_i \geq 1\}} = 0,$$

(82)

for all $i$. This reduces (79) to

$$\frac{d}{dt} \Pr(\tilde{K}(t) = k)\bigg|_{t=0} = \sum_i u_i \pi(k - e_i) \mathbbm{1}_{\{k_i \geq 1\}} - \sum_i \frac{d}{dt} \mathbb{E}[F(k, k + e_i, t)]\bigg|_{t=0} + \sum_i (k_i + 1) \mu_i \pi(k + e_i) + \sum_{i,i'} (k_i + 1) \mu_{i,i'} \pi(k + e_i - e_i') \mathbbm{1}_{\{k_i \geq 1\}} - \sum_i k_i \mu_i \pi(k) - \sum_{i,i'} k_i \mu_{i,i'} \pi(k),$$

(83)
Now we discuss based on whether \( \pi(k) = 0 \). If \( \pi(k) \neq 0 \), by the definition of our policy, for all \( i \),

\[
\frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0} = \frac{u_i(k)}{\pi(k)} \cdot \mathbb{P}(K(0) = k) = u_i(k). \tag{84}
\]

which is (80). Combining the above equation and the stationary equation (32) satisfied by \( (\pi, (u_i)_{i \in I}) \), we conclude that the RHS of (83) is zero, i.e.

\[
\frac{d}{dt} \mathbb{P}(K(t) = k) \bigg|_{t=0} = 0,
\]

which is (81). For the case when \( \pi(k) = 0 \) and \( \sum_i u_i(k) \neq 0 \), because the system immediately leave the configuration \( k \) after reaching it through a nominal transition,

\[
\frac{d}{dt} \mathbb{P}(K(t) = k) \bigg|_{t=0} = 0,
\]

i.e., the LHS of (83) is 0. Again we compare (83) against the stationary equation (32) and get

\[
\sum_i \frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0} = \sum_i u_i(k).
\]

By the definition of our policy, we have

\[
\frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0} = \frac{u_i(k)}{\sum_i u_i(k)} \cdot \sum_i \frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0} = u_i(k). \tag{85}
\]

which is (80). For the case when \( \pi(k) = 0 \) and \( \sum_i u_i(k) = 0 \), (32) implies that \( u_i(k - e_i) = 0, \pi(k + e_i) = 0, \pi(k + e_i - e_i') = 0 \) for any \( i \). Then (83) is further reduced to

\[
\frac{d}{dt} \mathbb{P}(K(t) = k) \bigg|_{t=0} = -\sum_i \frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0}.
\]

Because \( \mathbb{P}(K(t) = k) \geq 0 \) and \( \mathbb{P}(K(0) = k) = 0 \), the LHS of the above expression is non-negative. However, the RHS of the above expression is non-positive. Therefore, both sides equal to zero, thus we have

\[
\frac{d}{dt} \mathbb{P}(K(t) = k) \bigg|_{t=0} = 0 \quad \text{and} \quad \frac{d}{dt} \mathbb{E} \left[ F(k, k + e_i, t) \right] \bigg|_{t=0} = u_i(k).
\]

Having proved the base case, we do the induction step. Suppose we have proved (80) and (81) for all \( k \) such that \( \sum_{i \in I} k_i \leq m - 1 \) for some integer \( m \geq 1 \). We consider \( k \) with \( \sum_{i \in I} k_i = m \). By the induction hypothesis,

\[
\frac{d}{dt} \mathbb{E} \left[ F(k - e_i, k, t) \right] \bigg|_{t=0} = u_i(k - e_i) \mathbb{1}_{\{k_i \geq 1\}}. \tag{86}
\]

Then we repeat the arguments after (82) of the base case verbatim. By induction, we have proved (80) and (81).

Therefore, given the policy and initial distribution, the distribution of \( K(t) \) is stationary, i.e., we always have \( \mathbb{P}(K(t) = k) = \pi(k) \) for all \( k \in \mathcal{K} \). As a result, an analogue of (80) holds for all \( t \geq 0 \): \( \mathbb{E}[F(k, k + e_i, t)] \) is differentiable with respect to \( t \) and

\[
\frac{d}{dt} \mathbb{E}[F(k, k + e_i, t)] = u_i(k),
\]

for all \( k \in \mathcal{K} \) and all \( i \in I \). Therefore,

\[
\lim_{T \to \infty} \frac{1}{T} \int_0^T \mathbb{P}(K(t) = k) dt = \lim_{T \to \infty} \frac{1}{T} \cdot \pi(k) \cdot T = \pi(k),
\]

\[
\lim_{T \to \infty} \frac{1}{T} \mathbb{E}[F(k, k + e_i, T)] = \lim_{T \to \infty} \frac{1}{T} \int_0^T \frac{d}{dt} \mathbb{E}[F(k, k + e_i, t)] dt = \lim_{T \to \infty} \frac{1}{T} \cdot u_i(k) \cdot T = u_i(k).
\]
This completes the proof. \(\Box\)

B.2 Proof of Theorem 4

**Proof.** By Lemma 8, under the policy \(\bar{\sigma}(\pi^*, (u^*_i)_{i \in I})\), \(\pi^*\) is a stationary distribution, and \((u^*_i)_{i \in I}\) are the corresponding transition frequencies. Recall the single-server optimization \(\overline{P}((\lambda_i r)_{i \in I}, \epsilon)\) problem

\[
\begin{align*}
\text{minimize} & \quad \overline{N} \\
\text{subject to} & \quad \mathbb{E} \left[ h(\overline{K}(\infty)) \right | \overline{K}(\infty) \neq 0] \leq \epsilon, \\
& \quad \overline{N} \cdot \bar{\lambda}_i = \lambda_i r, \quad \forall i \in I. 
\end{align*}
\]

(3 revisited)

Observe that under the policy \(\bar{\sigma}(\pi^*, (u^*_i)_{i \in I})\), the cost of resource contention is \(h^T \pi \leq \epsilon(1 - \pi_0)\), the request rate of type \(i\) jobs is \(\bar{\lambda}_i = 1^T_0 u^*_i = \Phi^* \cdot \lambda_i\), so

\[
\mathbb{E} \left[ h(\overline{K}(\infty)) \right | \overline{K}(\infty) \neq 0] = \frac{h^T \pi}{1 - \pi_0} \leq \epsilon,
\]

\[
\bar{\lambda}_i = \Phi^* \cdot \lambda_i, \quad \forall i \in I.
\]

where we have used the fact that \(h(0) = 0\) in the first equality. Therefore, \((\Phi^*/r, \bar{\sigma}(\pi^*, (u^*_i)_{i \in I}), \pi)\) is a feasible solution to the single-server problem \(\overline{P}((\lambda_i r)_{i \in I}, \epsilon)\), achieving the objective value of \(r/\Phi^*\), which is the optimal value because \(r/\Phi^* \leq \overline{N}^*\). \(\Box\)