arXiv:1912.09870v2 [cs.PF] 3 Mar 2023

QoS-aware energy-efficient workload routing and server speed control policy in data centers: a robust queueing theoretic approach

Seung Min Baik [a] and Young Myoung Ko [a]

[a] Department of Industrial and Management Engineering, Pohang University of Science and Technology, Pohang, Gyeongbuk, 37673, South Korea

Abstract

Operating cloud service infrastructures requires high energy efficiency while ensuring a satisfactory service level. Motivated by data centers, we consider a workload routing and server speed control policy applicable to the system operating under fluctuating demands. Dynamic control algorithms are generally more energy-efficient than static ones. However, they often require frequent information exchanges between routers and servers, making the data centers' management hesitate to deploy these algorithms. This study presents a static routing and server speed control policy that could achieve energy efficiency similar to a dynamic algorithm and eliminate the necessity of frequent communication among resources. We take a robust queueing theoretic approach to response time constraints for the quality of service (QoS) conditions. Each server is modeled as a $G/G/1$ processor sharing queue, and the concept of uncertainty sets defines the domain of stochastic primitives. We derive an approximative upper bound of sojourn times from uncertainty sets and develop an approximative sojourn time quantile estimation method for QoS. Numerical experiments confirm the proposed static policy offers competitive solutions compared with the dynamic algorithm.

Keywords: data center, cloud server cluster, static control policy, load balancing, workload routing, server speed control, $G/G/1/PS$ queue, robust queueing theory, sojourn time quantile estimation.

1 Introduction

The demand for computing power has been skyrocketing to process exploding data from mobile devices and associated cloud services. Global ICT companies such as Amazon, Google, and Facebook have been aggressively building large-scale data centers to fulfill the demand for computing power while pursuing energy efficiency for offering services with low carbon emissions. Currently, international ICT firms believe that data centers should contribute to improving environmental sustainability (Hölzle, 2020), which is one of the most critical necessities for the survival of humanity.

A recent report by Google (Google, 2019) states that designing efficient data centers is one of its top priorities. Data centers consumed approximately 200 TWh of electricity in 2018, that is, approximately 1% of the total electricity consumption worldwide in 2018 (Jones *et al.*, 2018). Cisco, a global ICT company, expected that network traffic and workload in data centers will increase by 80% and 50%, respectively, by 2021; this implies a rapid increase in energy consumption (Cisco, 2018). However, current data center operations are inefficient, and average CPU utilization remains low at around 20% (Jiang *et al.*, 2019; Cortez *et al.*, 2017). Energy efficiency can be significantly improved by reducing energy-inefficient operations. As servers and cooling systems account for more than 80% of energy consumption in a data center (Pelley *et al.*, 2009; Rong *et al.*, 2016), we focus on server energy efficiency.

The studies on energy efficiency in data centers approach the problem at two levels: planning and operation. Studies at the planning level focused on designing data centers through static resource allocation, such as server virtualization and server clustering. Once decided, modifying the setting in real time is not easy. Xiong *et al.* (2019) formulated an optimization model for server clustering and proposed a metaheuristics algorithm to solve the mixed-integer non-linear programming problem. Urgaonkar *et al.* (2007), Gallego Arrubla *et al.* (2013), and Cho and Ko (2018) also addressed the planning-level problem in different ways. Under the context established by the planning level, research at the operation level examined the optimal load balancing and server speed scaling to process real-time requests. Because of the real-time nature of operations, *dynamic* load balancing and control algorithms have been considered in most of the energy efficiency studies at this level. Cho and Ko (2020) studied performance stabilization in a processor sharing queue with a time-varying traffic pattern considering the mean virtual response time. George and Harrison (2001) and Ata and Shneorson (2006) proposed control methods for single station systems based on quantitative mathematics such as Markov decision processes and queueing theory. Ko and Cho (2014), Cho and Ko (2022), and Andrew *et al.* (2009) proposed heuristic control methods. Bilal *et al.* (2015) developed a power-aware scheduling algorithm to simultaneously reduce makespan and energy consumption. Leading ICT companies, including Google and Amazon, are also utilizing artificial intelligence and machine learning techniques (Google, 2019; Amazon, 2019).

Practically, however, implementing dynamic algorithms for a load balancer and servers is challenging. Dynamic algorithms, including distributed ones, often require frequent communications

between network resources (e.g., routers and servers manufactured by different vendors and running heterogeneous operating systems to share their current status). Even if the network resources are equipped or retrofitted with a *universal* software enabling all those communications, the management in charge of operation may be more concerned with the reliability of their service and reluctant to take a risk by adopting dynamic control algorithms. When we contacted a company operating several large data centers in Korea, the manager initially expressed interest but finally declined the use of dynamic algorithms for the aforementioned reason.

In this regard, we investigate a static control policy that does not require frequent updates of the real-time status of resources; we determine the routing probabilities and server speeds once in a while, and resources autonomously run according to the policy. We use the terms *control* and *scaling* interchangeably throughout the study. Static control methods are generally less efficient than dynamic ones, particularly for situations with high variance and load fluctuations (Chandra and Sahoo, 2009; Wierman *et al.*, 2009). Chen and Li (2015), who investigated the interaction between load balancing and speed scaling, also pointed out that heterogeneity within a system is the main cause of inefficiency regardless of the number of servers.

To overcome this drawback, we attempt to develop a static control policy that can compete with the dynamic algorithm in terms of performance. We also explore situations under which the static policy works well. The term *static* does not indicate that the problem is solved only once and then used indefinitely. Depending on the changes in traffic patterns, operators can solve the problem again and update the routing probabilities and server speeds. The difference in the dynamic control is based on how often the control decisions are made: *real time or not*.

Previous studies at the operation level considering the quality of service (QoS) focused only on task scheduling (routing) or slack reclamation (speed scaling) (Rizvandi and Zomaya, 2012; Uddin *et al.*, 2015). Various QoS metrics were handled in studies in which task scheduling and slack reclamation were considered simultaneously; however, most of these studies involved non-probabilistic hard constraints in the model (Scarpiniti *et al.*, 2018; Gul *et al.*, 2016). Probabilistic QoS constraints were handled in only a few studies; however, the QoS related to the response time was rarely considered (Karpowicz, 2016). For example, Baccarelli *et al.* (2017) proposed a dynamic network flow control (routing) and virtual machine frequency scaling algorithm. Their QoS is related to the server's utility or processing capability from the operator's perspective.

However, Ko and Cho (2014) employed the solution of an optimization problem for workload routing and server speed scaling while considering sojourn time-related probabilistic QoS constraints. Their algorithm successfully reflected the real-time status of resources using an iterative method converging to the optimal solution of the problem. One of the most crucial advantages of this approach was that it only required communications between the load balancer and each server, not inter-server communications, which enabled distributed control. Their dynamic algorithm satisfied the response time constraints regarding the service level agreements; however, the solution appeared significantly conservative as it used a loose upper bound for the constraints. For instance, as we will see later in Section 4, the average delay probability with given threshold 5 from the simulation is 0.000034 (i.e., $\mathbb{P}(S \geq 5) \approx 0.000034$) when the response time constraint is set to be $\mathbb{P}(S \geq 5) \leq 0.05$—delay time threshold 5 with violation probability 0.05.

Therefore, we aim to derive a static policy with a tighter bound using robust queueing theory rather than the heavy-traffic approximation by Ko and Cho (2014). We describe the difference between the two approaches in detail and justify the reason for adopting a robust queueing theoretic approach in Section 3.1. Our static policy routes an incoming request to a server through a fixed probabilistic rule without real-time communications and operates the server at a constant speed when serving workload; a server runs at the minimum speed when idle.

Based on this information, the contributions of this study can be summarized as follows:

- The proposed static control policy satisfies the response time constraints less conservatively with more competitive energy efficiency than the existing dynamic control algorithm.
- This is the first study in which robust queueing theory is applied to processor sharing queues ($G/G/1/PS$ queues) suitable for modeling the CPU of a server using a time-slicing policy.
- We introduce new uncertainty sets for the processor sharing (PS) discipline by extending previous results for the first-come-first-served (FCFS) queue (Bandi *et al.*, 2015; Whitt and You, 2018, 2019). Additionally, we propose a methodology for determining variability parameters in PS queues to match the size of uncertainty sets to the target probability.

The remainder of this study is organized as follows. Section 2 describes the system settings and the formulation of an optimization problem for obtaining the static control policy. Section 3 briefly introduces the robust queueing theory and explains the steps for deriving a tractable inequality

that approximates the response (sojourn) time constraint. Section 4 presents numerical experiments conducted by comparing the performances of the proposed static policy and the dynamic control algorithm; it also presents a discussion on the results related to the static policy. Section 5 provides conclusions and suggestions for the future research.

2 Problem description

Section 2.1 describes the system with the probabilistic routing scheme. Section 2.2 constructs an optimization model that minimizes energy consumption under QoS conditions concerning response time.

2.1 *System description*

We consider a system consisting of I applications and J servers with a router, as shown in Figure 1. The individual servers can have different power functions and handle various applications. The server clusters and corresponding application assignments (virtualization) are given. Routing service requests follow a time-invariant probabilistic rule with the probability matrix $P = [p_{ij}]$. The arrival process of each application follows a renewal process, where inter-arrival time and workload random variables follow some general distributions having finite means and variances. When a service request of an application arrives, the router (or the load balancer) assigns the request to one of the J servers according to the predefined probability. Each server processes all requests in a round-robin fashion at the same rate (i.e., we model each server as a PS queue). Thus, analyzing the thinning and superposition of independent renewal processes is necessary to deal with each server. Servers operate at different speeds when they are either busy or idle. When the server j is in a busy period, it runs at a predetermined speed x_j. Otherwise, it operates at minimum speed x_j^{min}. The QoS conditions are defined in terms of response time; the probability that the response time of a request–the time between arrival and service completion–exceeds a specific threshold must fall below a particular level.

Our model is a simplified representation of a real system that comprises a single router and servers with a single power function. The model, however, can be extended to multi-tier architecture or multi-core processors by suitably combining multiple models and adjusting power functions; this

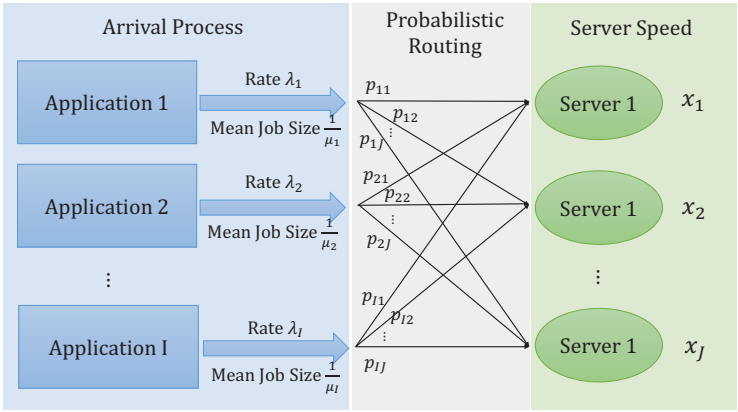


Figure 1: System model with probabilistic routing scheme

can be explored in future work.

2.2 *Optimization model*

The optimization model's objective is to minimize servers' energy consumption in a data center. The decision variables are routing probability matrix $P = [p_{ij}]$ and servers' speed x_j. We write the optimization model denoted by (M1) as follows:

$$\text{(M1)} \quad \text{Min} \sum_{j \in \mathcal{S}} \left(C_j(x_j)\rho_j + C_j\left(x_j^{min}\right)(1-\rho_j) \right)$$

$$\text{s.t. } 0 \le p_{ij} \le 1, \quad \forall i \in \mathcal{A}, \forall j \in \mathcal{S}_i, \tag{2.1}$$

$$\sum_{j \in \mathcal{S}_i} p_{ij} = 1, \quad \forall i \in \mathcal{A}, \tag{2.2}$$

$$x_j^{min} \le x_j \le x_j^{max}, \quad \forall j \in \mathcal{S}, \tag{2.3}$$

$$\mathbb{P}(S_j \ge \delta_j) \le \varepsilon_j, \quad \forall j \in \mathcal{S}. \tag{2.4}$$

In the objective function, $C_j(\cdot)$ denotes the power function of server j. We consider long-run average power consumption under a static control policy. The fraction of the busy period for server j equals traffic intensity ρ_j because the PS discipline satisfies work-conserving law. Thus, single server's average power consumption is a weighted sum of power function values with probabilities that the server is busy or idle. The objective value of the optimization model is the total power consumption over the entire server set denoted by $\mathcal{S}$. The power function is a convex function of server speed (x_j) (e.g., a cubic function in Wierman *et al.* (2012)). We use a more generic cubic

representation, $C_j(x_j) = K_j + \alpha_j x_j^3$ as in Horvath *et al.* (2007). The power functions do not have to be a polynomial function and can be any convex function. Changing the objective function, however, does not affect the crucial parts of our result—construction of uncertainty sets and approximation of QoS constraints.

Constraint (2.1) is a basic probability proposition. As shown in Figure 1, each arriving request of application i is assigned to server j with a probability of p_{ij}. $\mathcal{A}$ is the set of all applications, and $\mathcal{S}_i$ refers to the set of servers having application i. Constraint (2.2) implies that the sum of the routing probabilities for each application should be one. Constraint (2.3) describes the feasible speed range of the servers. Constraint (2.4) is the direct representation of the QoS conditions, stating that the probability of the steady-state response time of server j, S_j, exceeding the threshold δ_j should be lower than the small probability of ε_j. Constraint (2.4) sets a lower bound of server speed (or an upper bound of the response time) and requires a queueing approximation for tractability. Making a bound as tight as possible is critical because a loose bound increases server speed and causes unnecessary energy waste. Therefore, the approximation of constraint (2.4) is an essential part of this study.

Several other significant sources consume electricity, such as storage, memory, and network, other than CPUs. However, combining all those sources into a single model highly complicates the problem. Therefore, we would restrict our focus on workload routing and server speed scaling for CPU-intensive jobs with the QoS conditions, which bound the tail probability of response time.

3 QoS constraints

In this section, we reformulate the QoS conditions using robust queueing theory. Section 3.1 briefly explains the robust queueing theory. Section 3.2 describes how the robust queueing theoretic approach differs between FCFS and PS queues. Sections 3.3–3.5 describe the steps for deriving a tractable condition that approximates constraint (2.4). Section 3.3 constructs uncertainty sets suitable for PS queues, Section 3.4 derives the response time bound regarding the uncertainty sets, and Section 3.5 explains a procedure for choosing appropriate variability parameters. Sections 3.6 and 3.7 describe how the uncertainty sets vary for the thinning and superposition of arrival processes. Section 3.8 presents the analysis result for each server.

3.1 *Robust queueing theory*

Bandi *et al.* (2015) developed the robust queueing theory as an alternative approach to queueing analysis based on robust optimization. Traditional queueing theory has limitations such that even simple queueing systems are not tractable when the arrival processes or workloads are non-Markovian. To avoid this difficulty, the analysis of non-Markovian queues relies on simulation and approximation.

A large number of simulation runs can generate accurate results. However, each run requires a long time to produce an outcome, particularly in heavy-tailed systems. Approximation approaches can provide explicit mathematical expressions though they work well in some limited situations. Our study takes the approximation approach because we require analytical expressions to be embedded into the optimization model. A popular method, known as the heavy-traffic approximation, tends to underestimate the response time in our prestudy experiments, which causes violations of the QoS conditions. Therefore, we take a new approach—robust queueing theory—to construct robust and tractable expressions for constraint (2.4) while utilizing the findings of heavy-traffic approximation results; we take the functional form of the heavy-traffic approximation for some parameters, similar to Bandi *et al.* (2015) that uses the functional form of the Kingman's bound (Kingman, 1970).

The fundamental difference between the heavy-traffic approximation and robust queueing theory lies in the treatment of random variables. The heavy-traffic approximation first derives the distribution of the target performance measure, such as response time. Then, it calculates the quantile of the random variable using a cumulative distribution function. The robust queueing theory reverses this order. We compare the robust queueing theoretic approach to the heavy traffic approximation used by Ko and Cho (2014) that approximates the response time process as a reflected Brownian motion using the steady-state results for the $G/G/1/PS$ queue in Zhang and Zwart (2008):

$$S \approx \tilde{V}(\infty) \approx \beta \tilde{Z}(\infty),$$

where $\tilde{V}$ is a virtual response time and $\tilde{Z}$ is a reflected Brownian motion with drift θ/β_e and variance $\beta(C_a^2 + C_s^2)/\beta_e^2$. We refer the readers to Section 3 in Ko and Cho (2014) for a detailed explanation of other parameters. The approximate probability of exceeding a threshold, δ, is given by:

$$\mathbb{P}(S \geq \delta) \approx \exp\left(-\frac{\theta(1+C_s^2)\delta}{\beta\,(C_a^2+C_s^2)}\right).$$

The QoS constant ε is treated at the last moment.

In the robust queueing theoretic approach, however, ε appears at the beginning of the analysis. We first restrict the probabilistic region for stochastic primitives and then proceed with the analysis. For better understanding, we defer the precise mathematical details about the definition of uncertainty sets' probability and arrival index on purpose to Sections 3.2–3.4. We only depict an abstract concept here. Suppose we define the uncertainty sets of stochastic primitives, inter-arrival times $\mathbf{T}(n) = (T_1, T_2, \ldots, T_n)$, and workloads $\mathbf{X}(n) = (X_1, X_2, \ldots, X_n)$, with probability $1 - \varepsilon$ as follows:

$$\mathbb{P}\left(\mathbf{T}(n) \in \mathcal{U}^a(n), \mathbf{X}(n) \in \mathcal{U}^s(n)\right) = 1 - \varepsilon.$$

We derive an approximative upper bound of the worst-case sojourn time among the uncertainty sets of $\mathbf{T}(n)$ and $\mathbf{X}(n)$ as follows:

$$\max_{\mathbf{T}(n)\in\mathcal{U}^a(n),\mathbf{X}(n)\in\mathcal{U}^s(n)} S_n(\mathbf{T}(n), \mathbf{X}(n)) \leq S_{\mathrm{UB}}, \ \forall n,$$

and use it to construct a tractable condition that can replace the QoS constraint, that is, we substitute $\mathbb{P}(S \geq \delta)$ with $S_{\mathrm{UB}} \leq \delta$. Considering ε in advance is essential in constructing uncertainty sets. Adopting a robust queueing theory can provide analytical representations with reasonable accuracy and ensure that the QoS conditions are satisfied not too conservatively.

3.2 *Comparison of FCFS and PS queues*

Before explaining the derivation of the worst-case sojourn time and its bound, we review the robust queueing analysis for FCFS queues and determine what we should modify under PS discipline. Previous studies on robust queueing theory (Bandi *et al.*, 2015; Whitt and You, 2018, 2019) follow a similar procedure for response time analysis. We can break it down into the following four steps:

1. Construct uncertainty sets $\mathcal{U}^a(n)$ and $\mathcal{U}^s(n)$ for n inter-arrival time and workload random variables $\mathbf{T}(n) = (T_1, T_2, \ldots, T_n)$ and $\mathbf{X}(n) = (X_1, X_2, \ldots, X_n)$.

2. Express the exact response time of the n^{th} arriving job, $S_n(\mathbf{T}(n), \mathbf{X}(n))$, via stochastic primitives $\mathbf{T}(n)$ and $\mathbf{X}(n)$.

3. Calculate the worst-case response time of the n^{th} arriving job with respect to the stochastic primitives within the uncertainty sets, $\hat{S}_n(\mathcal{U}^a(n), \mathcal{U}^s(n)) = \max_{\mathbf{T}(n)\in\mathcal{U}^a(n),\mathbf{X}(n)\in\mathcal{U}^s(n)} S_n(\mathbf{T}(n), \mathbf{X}(n))$.

4. Derive a bound S_{UB} that is independent of n and is an upper bound of $\hat{S}_n(\mathcal{U}^a(n),\mathcal{U}^s(n))$ for all n.

In steps 1 to 3, the target performance measure is bound by the desired probability; however, the bound is valid only for the request with a specific (finite) arrival index. Thus, step 4 generalizes the result for the arbitrary arrival index and, consequently, culminates in the steady-state response time.

However, we cannot directly apply the above approach to a PS queue. For a $G/G/1/FCFS$ queue, we can represent the exact expression of the response time in step 2 using a linear form of stochastic primitives. However, for a $G/G/1/PS$ queue, the expression of the PDE-form is only available. Equations (3.1) and (3.2) represent the response time expressions in the $G/G/1/FCFS$ and $G/G/1/PS$ queues, respectively. For a FCFS queue, by the Lindley recursion (Lindley, 1952),

$$S_n^{FCFS} = \max_{1\le k\le n}\left(\sum_{i=k}^{n} X_i - \sum_{i=k}^{n-1} T_i\right), \tag{3.1}$$

where S_n^{FCFS}, X_n, and T_n denote the response time of the n^{th} arriving job under the FCFS discipline, the workload of the n^{th} arriving job, and the inter-arrival time between the n^{th} and the $(n+1)^{\text{th}}$ arrivals, respectively. For a PS queue, we have

$$\begin{aligned} A_n &= \sum_{i=1}^{n} T_i, \quad D_n = A_n + S_n^{PS}, \quad X_n = \int_{A_n}^{D_n} \frac{1}{N(t)} dt, \\ N(t) &= \sum_{n=1}^{\infty} \mathbb{1}_{\{A_n\le t<D_n\}} = \sum_{n=1}^{\infty} \mathbb{1}_{\{A_n\le t\}} - \sum_{n=1}^{\infty} \mathbb{1}_{\{D_n\le t\}} \end{aligned} \tag{3.2}$$

where S_n^{PS}, $N(t)$, A_n, and D_n denote the response time of the n^{th} arriving job under the PS discipline, the number of unfinished jobs in the queue at time t, and the arrival and departure time of the n^{th} arriving job, respectively. The rate at which the remaining workload of each job decreases is inversely proportional to $N(t)$ for a PS queue. We cannot express the response time in a linear form of X_n and T_n.

Therefore, we cannot directly follow the same steps 1 to 4 to analyze a PS queue. The optimization problem is almost intractable if we embed equation (3.2) as a model constraint. To avoid the PDE-form, we integrate steps 2 and 3 in the robust queueing analysis procedure by creating a loose worst-case upper bound instead of seeking an alternative near-exact expression or a heuristic solution. Although our approach loosens the inequality and produces more conservative

results, the experimental results in Section 4 demonstrate that our method successfully guarantees constraint (2.4) without being overly conservative.

3.3 *Uncertainty sets*

We adopt a similar approach to Bandi *et al.* (2015) and Whitt and You (2018) in constructing uncertainty sets. Here, we apply the central limit theorem (CLT), that is, for independent and identically distributed (i.i.d.) random variables ξ_i with the same distribution as ξ, the quantity $(\sum_{i=1}^{n} \xi_i - n\mathbb{E}[\xi])/\sqrt{n\text{Var}[\xi]}$ becomes asymptotically standard normal as n approaches infinity. To apply the CLT, this study assumes the random variables that characterize the arrival process (i.e., inter-arrival time and workload) to have a finite mean and variance, which is a reasonable assumption for practical purposes. The worst-case response time is achieved when the inter-arrival times are short and the workloads are large. Therefore, we construct the uncertainty sets with one-sided inequalities to consider situations with shorter inter-arrival times and larger workloads.

We construct the uncertainty set of inter-arrival times, $\mathcal{U}^a(n)$ as follows:

$$\mathcal{U}^a(n) = \left\{ \mathbf{T}(n) = (T_1, T_2, \ldots, T_n) \,\middle|\, \frac{\sum_{i=k}^{n} T_i - \frac{n-k+1}{\lambda}}{\sqrt{n-k+1}} \geq -\Gamma_a,\ \forall k, 1 \leq k \leq n \right\}. \tag{3.3}$$

Here, $1/\lambda$ and σ_a are the mean $\mathbb{E}[T_i]$ and standard deviation $\sqrt{\text{Var}[T_i]}$ of the inter-arrival times, respectively, and Γ_a is a constant called variability parameter related to the size of the uncertainty set or its corresponding probability. Similarly, we construct the uncertainty set of service workloads, $\mathcal{U}^s(n)$, as follows:

$$\begin{aligned} \mathcal{U}^s(n) = \Bigg\{ \mathbf{X}(n) = (X_1, X_2, \ldots, X_n) \,\Bigg|\, & \frac{\sum_{i=k}^{n} X_i - \frac{n-k+1}{\mu}}{\sqrt{n-k+1}} \leq \Gamma_s,\ \forall k, 1 \leq k \leq n \\ & \text{and}\ \ \frac{\sum_{i=k}^{n-1} X_i - \frac{n-k}{\mu}}{\sqrt{n-k}} \leq \Gamma_s,\ \forall k, 1 \leq k \leq n-1 \Bigg\}. \end{aligned} \tag{3.4}$$

Here, $1/\mu$ and σ_s are the mean $\mathbb{E}[X_i]$ and standard deviation $\sqrt{\text{Var}[X_i]}$ of the workloads, respectively. Γ_s is a variability parameter for the uncertainty set of service workloads.

The uncertainty set for service workloads $\mathcal{U}^s(n)$ differs from the one defined by Bandi *et al.* (2015). The sojourn time can be expressed using the terms of finite service workload sums, $\sum_{i=k}^{n} X_i$, in an FCFS queue. The sojourn time in a PS queue, however, cannot be explicitly expressed with stochastic primitives; it can only be bounded via terms of $\sum_{i=k}^{n} X_i + \sum_{i=k}^{n-1} X_i$, and the

derivation procedure is explained in detail in Section 1.1 in the online supplement. Thus, the modified uncertainty set structure bounds above both $\sum_{i=k}^{n} X_i$ and $\sum_{i=k}^{n-1} X_i$.

3.4 *Response time bound*

This section derives the bound of the response time from the uncertainty sets defined in equations (3.3) and (3.4). Here, we assume that variability parameters Γ_a and Γ_s are predetermined to appropriate values such that the size of uncertainty sets corresponds to the target probability level $1-\varepsilon$. Finding these Γ_a and Γ_s is explained in the following section. Unlike FCFS queues, we cannot express the response time and its worst-case value within the uncertainty sets in an exact form of stochastic primitives for PS queues. Therefore, our methodology constructs an inequality for response time and directly obtains its upper bound. This procedure merges steps 2 and 3 in Section 3.2.

Based on Theorem 3.1 of Brandt and Brandt (2006), the following holds for the steady-state sojourn (response) time random variables, S^{PS} and S^{FCFS}, of $G/G/1/PS$ and $G/G/1/FCFS$ systems, respectively, that share the same stochastic primitives:

$$\mathbb{E}[S^{PS}] + \mathbb{E}[X] \leq 2\mathbb{E}[S^{FCFS}].$$

Because we are dealing with a probabilistic QoS constraint, $\mathbb{P}(S \geq \delta) \leq \varepsilon$, in *steady state*, we use the following inequality as an approximation:

$$S_n^{PS} + X_n \leq 2S_n^{FCFS}, \tag{3.5}$$

where S_n^{FCFS} and S_n^{PS} are the sojourn times of the n^{th} arriving job under FCFS and PS disciplines, respectively. For a large enough n, we can think the queue is in steady state. Assuming this inequality, the worst-case sojourn time with respect to the stochastic primitives within uncertainty sets is bounded as follows (See Section 1.1 in the online supplement for the proof):

$$\hat{S}_n^{PS}(\mathcal{U}^a(n),\mathcal{U}^s(n)) = \max_{\mathbf{T}(n)\in\mathcal{U}^a(n),\mathbf{X}(n)\in\mathcal{U}^s(n)} S_n^{PS}(\mathbf{T}(n),\mathbf{X}(n)) \leq S_{\text{UB}} := \frac{(\Gamma_a+\Gamma_s)^2\lambda}{2(1-\rho)} + \frac{2-\rho}{\lambda}. \tag{3.6}$$

S_{UB} is a valid upper bound regardless of n, which only relies on the characteristics of the arrival process and the size of the uncertainty sets. We note that the derivation draws on approximation; the selection of variability parameters approximately regulates the tightness of the bound.

Section 3.5 describes a procedure to find appropriate parameter values so that the following inequality about steady-state sojourn time random variable S^{PS} (we drop the superscript PS now for notational convenience) holds not too conservatively:

$$\mathbb{P}\left(S^{PS} \leq \frac{(\Gamma_a + \Gamma_s)^2 \lambda}{2(1-\rho)} + \frac{2-\rho}{\lambda}\right) \geq 1 - \varepsilon.$$

Therefore, we replace the QoS constraint with the following condition to deduce a heuristic policy:

$$S_{\text{UB}} = \frac{(\Gamma_a + \Gamma_s)^2 \lambda}{2(1-\rho)} + \frac{2-\rho}{\lambda} \leq \delta. \tag{3.7}$$

Given an arrival process, server speed scaling only affects workload-related parameters Γ_s, μ, or indirectly, ρ, in the aforementioned inequality.

The variability parameter values bridge the uncertainty sets and the associated probability. Bandi *et al.* (2015) presented the procedure for determining the values as *"translating the stochastic primitive data into robust primitive data, namely uncertainty sets with appropriate variability parameters."* The following section describes our approach (i.e., such a procedure for PS queues) in detail. We only state here that the variability parameters do not have to rely on n because we consider a large n to approximate steady state.

3.5 *Variability parameters*

This section describes how we determine the variability parameters to match the size of uncertainty sets with the target probability level under the PS discipline. The approximative upper bound S_{UB} derived in Section 3.4 is calculated from parameters of an arrival process $(\mu, \lambda, \sigma_a, \sigma_s)$ and variability parameters (Γ_a, Γ_s). Thus, given the arrival process, selecting the appropriate variability parameters is crucial for meaningful robust queueing analysis. Previous works (Bandi *et al.*, 2015; Whitt and You, 2018, 2019) also highlighted such importance and provided different methods for determining the variability parameter values. Among them, we adopt a similar approach to Bandi *et al.* (2015). They designed a functional form of variability parameters by considering the qualitative insights given by Kingman's bound (Kingman, 1970) and fitted the function coefficients using the simulation. Likewise, we follow a similar procedure while utilizing the findings of heavy-traffic approximation results. For this, it is important to ensure that the result of the robust queueing theoretic approach shares the same qualitative insights as the result of the traditional probabilistic framework.

In this work, we utilize the tail probability approximation of Zhang and Zwart (2008) for the steady-state sojourn time in a PS queue as follows:

$$\mathbb{P}(S \geq \delta) \approx \exp\left(-\frac{1+C_s^2}{C_a^2+C_s^2}\frac{1-\rho}{\rho}\lambda\delta\right) \leq \exp\left(-\frac{1}{C_a^2+C_s^2}\frac{1-\rho}{\rho}\lambda\delta\right).$$

Motivated by this approximation, we derive an approximative bound on the sojourn time quantile concerning the target probability level $1-\varepsilon$ as follows:

$$\delta \leq -\left(C_a^2+C_s^2\right)\frac{\rho}{(1-\rho)\lambda}\log\mathbb{P}(S \geq \delta) = \frac{\left(\rho\sigma_a^2+\frac{\sigma_s^2}{\rho}\right)\lambda}{1-\rho}\left(-\log\varepsilon\right). \tag{3.8}$$

Correspondingly, we consider the functional form of the variability parameters Γ_a and Γ_s as follows:

$$\begin{aligned}
\Gamma_a &= f_a(\mu,\lambda,\sigma_a,\sigma_s;1-\varepsilon) := \sigma_a,\\
\Gamma_s &= f_s(\mu,\lambda,\sigma_a,\sigma_s;1-\varepsilon) := \left(\theta_0+\theta_1\rho\sigma_a^2+\theta_2\frac{\sigma_s^2}{\rho}+\theta_3\frac{(1-\rho)(2-\rho)}{\lambda^2}\right)^{1/2}-\sigma_a,
\end{aligned} \tag{3.9}$$

where the traffic intensity is $\rho=\lambda/\mu$. This enables the approximative sojourn time upper bound, S_{UB}, to have functional dependence on $\lambda/(1-\rho)$ and $\rho\sigma_a^2, \sigma_s^2/\rho$ as the bound in equation (3.8) as follows:

$$S_{\mathrm{UB}} = \frac{(\Gamma_a+\Gamma_s)^2\lambda}{2(1-\rho)}+\frac{2-\rho}{\lambda} = \frac{\left(\theta_1\rho\sigma_a^2+\theta_2\frac{\sigma_s^2}{\rho}\right)\lambda}{2(1-\rho)}+\left(\frac{\theta_0\lambda}{2(1-\rho)}+\left(\frac{\theta_3}{2}+1\right)\frac{2-\rho}{\lambda}\right).$$

This is an analogue to the method for FCFS queues in Bandi *et al.* (2015), and we explain it briefly here for comparison. They utilize the Kingman's bound (Kingman, 1970) on the expected steady-state waiting time for $G/G/1/FCFS$ queues, $\mathbb{E}\left[W^{FCFS}\right]$, to bound the expected steady-state sojourn time as follows:

$$\mathbb{E}\left[S^{FCFS}\right] = \mathbb{E}\left[W^{FCFS}+X\right] = \mathbb{E}\left[W^{FCFS}\right]+\mathbb{E}\left[X\right] \leq \frac{(\sigma_a^2+\sigma_s^2)\lambda}{2(1-\rho)}+\frac{1}{\mu}.$$

They consider the functional form of the variability parameters as follows:

$$\begin{aligned}
\Gamma_a^{FCFS} &= f_a^{FCFS}(\rho,\sigma_a,\sigma_s) := \sigma_a,\\
\Gamma_s^{FCFS} &= f_s^{FCFS}(\rho,\sigma_a,\sigma_s) := (\theta_0+\theta_1\sigma_s^2+\theta_2\sigma_a^2\rho^2)^{1/2}-\sigma_a,
\end{aligned} \tag{3.10}$$

and state that the upper bound of the worst-case sojourn time by robust queueing theory (From Theorem 2) given by:

$$\frac{(\Gamma_a^{FCFS}+\Gamma_s^{FCFS})^2\lambda}{4(1-\rho)}+\frac{1}{\lambda} = \frac{(\theta_1\sigma_s^2+\theta_2\sigma_a^2\rho^2)\lambda}{4(1-\rho)}+\left(\frac{\theta_0\lambda}{4(1-\rho)}+\frac{1}{\lambda}\right)$$

Table 1: Notations for thinning of arrival process

Symbol	Description
λ_i	Arrival rate of application i
$\sigma_{a,i}$	Standard deviation of the inter-arrival time of application i
λ_{ij}	Arrival rate of application i into server j
$\sigma_{a,ij}$	Standard deviation of the inter-arrival time of application i into server j
$1/\mu_i$	Average workload of application i
$\sigma_{s,i}$	Standard deviation of the workload of application i
$P = [p_{ij}]$	Routing probability matrix

has same functional dependence on $\lambda/(1-\rho)$ and σ_a^2, σ_s^2 as Kingman's bound. They mention that their *"robust queueing theoretic approach leads to the same qualitative conclusions as stochastic queueing theory"*.

As done in Bandi *et al.* (2015), we first run simulations for a single server queue with different parameters of arrival process $(\lambda, \mu, \sigma_a, \sigma_s)$ and calculate the sojourn time quantile with predefined probability level $1-\varepsilon$. Because we generate instances with a large number of arrivals and services to be close to the steady state (we track S_n with sufficiently large n), the variability parameters do not have to consider index n. Then, we fit a function for the variability parameters in equation (3.9) such that the S_{UB} value is adapted to a simulation-based quantile value. We present the detailed procedure of the numerical experiments and results in Section 2 in the online supplement.

Once obtaining variability parameter values, we can substitute the QoS constraint with its approximate but tractable condition as in equation (3.7). However, the actual arrival process in our model is constructed through thinning and superposition of the external (original) arrival processes. The external arrivals of each application are probabilistically split or thinned by routing rules (thinning). This thinned process generates a flow of application i to server j, and a particular server receives an aggregate incoming request from the various applications (superposition). Therefore, to properly handle server-wise QoS constraints, we must examine the relationship between uncertainty sets of external, thinned, and aggregated processes. Section 3.6 presents a discussion on the thinning of arrival process, and Section 3.7 explains the superposition of arrival processes.

3.6 *Thinning of arrival process*

First, we investigate a thinned arrival process of application i entering server j. We could use the thinning operator in Theorem 6 in Bandi *et al.* (2015) as it provides a good approximation result

of the probabilistic routing policy. However, if the routing probability p_{ij} is small, it might lead to an overly optimistic result. Thus, we choose to construct the uncertainty sets of the thinned arrival process by directly deriving the mean and variance of its inter-arrival time and service workload random variables.

The server j can take the request of application i with probability p_{ij}. Let T_{ij} be a random variable denoting the inter-arrival time of application i to server j. We express T_{ij} using the random variable denoting the inter-arrival time of application i to the system, T_i, as follows:

$$T_{ij} = \sum_{n=1}^{N_{ij}} T_i^n = \begin{cases} T_i^1 & \text{if } N_{ij} = 1 \text{ with } \mathbb{P}(N_{ij} = 1) = p_{ij} \\ T_i^1 + T_i^2 & \text{if } N_{ij} = 2 \text{ with } \mathbb{P}(N_{ij} = 2) = (1 - p_{ij})p_{ij} \\ \vdots & \end{cases},$$

where T_i^n's are i.i.d. random variables with the same distribution as T_i and N_{ij} is a random variable denoting the number of external arrivals of application i until one application i arrival is routed to server j for the first time, i.e., N_{ij} is a geometric random variable with $\mathbb{P}(N_{ij} = n) = (1-p_{ij})^{n-1}p_{ij}$.

We derive the mean and variance of T_{ij} as follows (See Section 1.2 in the online supplement for the proof):

$$\mathbb{E}[T_{ij}] = \frac{\mathbb{E}[T_i]}{p_{ij}}, \quad \text{Var}[T_{ij}] = \frac{\text{Var}[T_i]}{p_{ij}} + (\mathbb{E}[T_i])^2 \frac{1 - p_{ij}}{p_{ij}^2}.$$

Thus, with the necessary notations defined in Table 1, the mean and variance of the inter-arrival time of the thinned process are:

$$\frac{1}{\lambda_{ij}} = \frac{1}{\lambda_i p_{ij}}, \quad \sigma_{a,ij}^2 = \frac{\sigma_{a,i}^2}{p_{ij}} + \frac{1 - p_{ij}}{\lambda_i^2 p_{ij}^2}.$$

The uncertainty set of the inter-arrival times of the thinned process is:

$$\mathcal{U}_{ij}^a(n) = \left\{ \mathbf{T}_{ij}(n) = (T_{ij}^1, T_{ij}^2, \ldots, T_{ij}^n) \,\middle|\, \frac{\sum_{l=k}^n T_{ij}^l - \frac{n-k+1}{\lambda_{ij}}}{\sqrt{n-k+1}} \geq -\Gamma_{a,ij},\ \forall k, 1 \leq k \leq n \right\}.$$

We can determine the variability parameter $\Gamma_{a,ij}$ with the derived mean $1/\lambda_{ij}$ and variance $\sigma_{a,ij}^2$ for the target probability level. The uncertainty set of the workloads of the thinned process is identical to the uncertainty set of workloads of application i as follows:

$$\begin{aligned} \mathcal{U}_{ij}^s(n) = \mathcal{U}_i^s(n) = \Bigg\{ \mathbf{X}_i(n) = (X_i^1, X_i^2, \ldots, X_i^n) \,\Bigg|\, & \frac{\sum_{l=k}^n X_i^l - \frac{n-k+1}{\mu_i}}{\sqrt{n-k+1}} \leq \Gamma_{s,i},\ \forall k, 1 \leq k \leq n \\ & \text{and } \frac{\sum_{l=k}^{n-1} X_i^l - \frac{n-k}{\mu_i}}{\sqrt{n-k}} \leq \Gamma_{s,i},\ \forall k, 1 \leq k \leq n-1 \Bigg\}. \end{aligned} \tag{3.11}$$

Table 2: Notations for superposition of arrival processes

Symbol	Description
$\bar{\lambda}_j$	Arrival rate for server j
$\bar{\sigma}_{a,j}$	Standard deviation of the inter-arrival time of arrival into server j
$1/\bar{\mu}_j$	Average workload for server j
$\bar{\sigma}_{s,j}$	Standard deviation of workload into server j

Similarly, we can determine the variability parameter $\Gamma_{s,i}$ with μ_i and $\sigma^2_{s,i}$ for the target probability level.

3.7 *Superposition of arrival processes*

Next, we aggregate multiple thinned processes to generate an actual aggregated arrival process to server j and derive its uncertainty sets. After superposition, the dependence on inter-arrival times might arise and invalidate the CLT assumptions, causing issues with the uncertainty set. However, to keep the structure of the CLT-based uncertainty set, we apply the method from Bandi *et al.* (2015) for constructing the uncertainty set for the inter-arrival times, which includes all the instances of the aggregate process' inter-arrival times. For the workloads, in which dependence issue does not arise, we derive the exact moments of the aggregate process' workload random variable and directly construct the uncertainty set.

First, we illustrate the set of inter-arrival times. In our model, all the uncertainty sets of inter-arrival times of the thinned processes to be merged are in CLT-based shape, that is, the term in the left-hand side of the inequality is inversely proportional to the square root of $n-k$ (or $n-k+1$). Therefore, we can apply the superposition operator in Theorem 5 of Bandi *et al.* (2015). For a more general design of uncertainty sets considering the heavy-tailed distributions and generalized CLT, we refer to Bandi *et al.* (2015) for details. The uncertainty set of inter-arrival times, $\bar{\mathcal{U}}^a_j(n)$, for the server-wise aggregated process is as follows:

$$\bar{\mathcal{U}}^a_j(n) = \left\{ \bar{\mathbf{T}}_j(n) = (\bar{T}^1_j, \bar{T}^2_j, \ldots, \bar{T}^n_j) \,\middle|\, \frac{\sum_{l=k}^{n} \bar{T}^l_j - \frac{n-k+1}{\bar{\lambda}_j}}{\sqrt{n-k+1}} \geq -\bar{\Gamma}_{a,j},\ \forall k, 1 \leq k \leq n \right\}. \tag{3.12}$$

This uncertainty set is slightly larger than the region where the aggregate process' inter-arrival times lie, as it considers the potential dependence between them. That is, the uncertainty set of the exact aggregate arrival intervals becomes the subset of the one in equation (3.12). With necessary

notations defined in Table 2, the mean arrival rate and variability parameter of the new uncertainty set are (See Section 1.3 in the online supplement for the proof):

$$\bar{\lambda}_j = \sum_{i\in\mathcal{A}_j} \lambda_{ij} = \sum_{i\in\mathcal{A}_j} \lambda_i p_{ij}, \quad \bar{\Gamma}_{a,j} = \frac{\left(\sum_{i\in\mathcal{A}_j}\left(p_{ij}C_{a,i}^2 + (1-p_{ij})\right)\right)^{1/2}}{\sum_{i\in\mathcal{A}_j}\lambda_i p_{ij}}, \tag{3.13}$$

where $C_{a,i}^2$ is a squared coefficient of variation of the inter-arrival time of application i.

Next, we describe the uncertainty set of workloads. From the results about the aggregated arrivals' workload in Section 7.1.2 of Gautam (2012), the mean and variance of the actual workload are:

$$\frac{1}{\bar{\mu}_j} = \frac{\sum_{i\in\mathcal{A}_j}\frac{\lambda_{ij}}{\mu_i}}{\sum_{i\in\mathcal{A}_j}\lambda_{ij}} = \frac{\sum_{i\in\mathcal{A}_j} p_{ij}\frac{\lambda_i}{\mu_i}}{\sum_{i\in\mathcal{A}_j} p_{ij}\lambda_i}, \quad \bar{\sigma}_{s,j}^2 = \frac{\sum_{i\in\mathcal{A}_j}\left(\lambda_{ij}\left(\sigma_{s,i}^2 + \left(\frac{1}{\bar{\mu}_j} - \frac{1}{\mu_i}\right)^2\right)\right)}{\sum_{i\in\mathcal{A}_j}\lambda_{ij}}. \tag{3.14}$$

With these exact moments of workload, the uncertainty set of service workloads, $\bar{\mathcal{U}}_j^a(n)$, for the server-wise aggregated process can be constructed as follows:

$$\begin{aligned}\bar{\mathcal{U}}_j^s(n) = \Bigg\{ \bar{\mathbf{X}}_j(n) = (\bar{X}_j^1, \bar{X}_j^2, \ldots, \bar{X}_j^n) \Bigg| & \frac{\sum_{l=k}^{n}\bar{X}_j^l - \frac{n-k+1}{\bar{\mu}_j}}{\sqrt{n-k+1}} \le \bar{\Gamma}_{s,j},\ \forall k, 1\le k\le n \\ \text{and}\quad & \frac{\sum_{l=k}^{n-1}\bar{X}_j^l - \frac{n-k}{\bar{\mu}_j}}{\sqrt{n-k}} \le \bar{\Gamma}_{s,j},\ \forall k, 1\le k\le n-1 \Bigg\}.\end{aligned} \tag{3.15}$$

As in equation (3.9), the variability parameter for workloads is:

$$\bar{\Gamma}_{s,j} = f_s\left(\bar{\mu}_j x_j, \bar{\lambda}_j, \bar{\sigma}_{a,j}, \frac{\bar{\sigma}_{s,j}}{x_j}; 1-\varepsilon_j\right). \tag{3.16}$$

Here, the workload mean and standard deviation are scaled according to the server speed because the derivation results in Section 3.5 assume unit service rate.

In the end, the uncertainty sets of the aggregated arrival process in equations (3.12) and (3.15) reflect both the characteristics of the external arrival process and routing policy. The variability parameters serve as a bridge between the external, thinned, and aggregated arrival processes. Thus, the relationship between the variability parameters of each process' uncertainty sets identifies the relationship between the processes via thinning and superposition.

3.8 *Server-wise analysis*

With the uncertainty sets of the actual arrival process into server j in equation (3.12), QoS constraints in the optimization model can be expressed in terms of the server speed scaling. Using the

results in equations (3.13)–(3.16), we construct a tractable condition that approximates the server j's response time constraint, $\mathbb{P}(S_j \geq \delta_j) \leq \varepsilon_j$, similar to equation (3.7) as follows:

$$\frac{(\bar{\Gamma}_{a,j} + \bar{\Gamma}_{s,j})^2 \bar{\lambda}_j}{2\left(1 - \frac{\bar{\lambda}_j}{\bar{\mu}_j x_j}\right)} + \frac{2 - \frac{\bar{\lambda}_j}{\bar{\mu}_j x_j}}{\bar{\lambda}_j} \leq \delta_j. \tag{3.17}$$

In equation (3.17), ρ is replaced by $\bar{\lambda}_j/\bar{\mu}_j x_j$ because we consider the scalable server speed instead of the unit speed at which unit workload is processed per unit time. The traffic intensity ρ_j is the server-wise mean arrival rate $\bar{\lambda}_j$ multiplied by the scaled server-wise mean workload $1/\bar{\mu}_j x_j$. Additionally, we calculate the workload variability parameter $\bar{\Gamma}_{s,j}$ considering the server speed in equation (3.17). There are no changes in the parameters related to the inter-arrival time.

Based on the derivation results so far, we construct a new optimization model (M2) as follows:

$$\text{(M2)} \quad \text{Min} \sum_{j \in \mathcal{S}} \left(C_j(x_j) \frac{\sum_{i \in \mathcal{A}_j} p_{ij} \frac{\lambda_i}{\mu_i}}{x_j} + C_j\left(x_j^{min}\right) \left(1 - \frac{\sum_{i \in \mathcal{A}_j} p_{ij} \frac{\lambda_i}{\mu_i}}{x_j}\right) \right)$$

$$\text{s.t. } 0 \leq p_{ij} \leq 1, \quad \forall i \in \mathcal{A}, \forall j \in \mathcal{S}_i, \tag{3.18}$$

$$\sum_{j \in \mathcal{S}_i} p_{ij} = 1, \quad \forall i \in \mathcal{A}, \tag{3.19}$$

$$x_j^{min} \leq x_j \leq x_j^{max}, \quad \forall j \in \mathcal{S}, \tag{3.20}$$

$$\bar{\lambda}_j = \sum_{i \in \mathcal{A}_j} \lambda_{ij} = \sum_{i \in \mathcal{A}_j} \lambda_i p_{ij}, \quad \forall j \in \mathcal{S}, \tag{3.21}$$

$$\bar{\sigma}_{a,j}^2 = \bar{\Gamma}_{a,j}^2 = \frac{\sum_{i \in \mathcal{A}_j} \left(p_{ij} C_{a,i}^2 + (1 - p_{ij})\right)}{\left(\sum_{i \in \mathcal{A}_j} \lambda_i p_{ij}\right)^2}, \quad \forall j \in \mathcal{S}, \tag{3.22}$$

$$\frac{1}{\bar{\mu}_j} = \frac{\sum_{i \in \mathcal{A}_j} p_{ij} \frac{\lambda_i}{\mu_i}}{\sum_{i \in \mathcal{A}_j} p_{ij} \lambda_i}, \quad \forall j \in \mathcal{S}, \tag{3.23}$$

$$\bar{\sigma}_{s,j}^2 = \frac{\sum_{i \in \mathcal{A}_j} \lambda_{ij} \left(\sigma_{s,i}^2 + \left(\frac{1}{\bar{\mu}_j} - \frac{1}{\mu_i}\right)^2\right)}{\sum_{i \in \mathcal{A}_j} \lambda_{ij}}, \quad \forall j \in \mathcal{S}, \tag{3.24}$$

$$\bar{\Gamma}_{s,j} = f_s\left(\bar{\mu}_j x_j, \bar{\lambda}_j, \bar{\sigma}_{a,j}, \frac{\bar{\sigma}_{s,j}}{x_j}; 1 - \varepsilon_j\right), \quad \forall j \in \mathcal{S}, \tag{3.25}$$

$$\frac{(\bar{\Gamma}_{a,j} + \bar{\Gamma}_{s,j})^2 \bar{\lambda}_j}{2\left(1 - \frac{\bar{\lambda}_j}{\bar{\mu}_j x_j}\right)} + \frac{2 - \frac{\bar{\lambda}_j}{\bar{\mu}_j x_j}}{\bar{\lambda}_j} \leq \delta_j, \quad \forall j \in \mathcal{S}. \tag{3.26}$$

The objective function of (M2) is the same as that of (M1). Constraints (3.18)–(3.20) are the same as constraints (2.1)–(2.3) in the optimization model (M1). The rest of the constraints in the

Table 3: Arrival process setting

Application type (i)	Inter-arrival time random variable			Workload random variable			Instant demand rate
	Distribution	Mean ($1/\lambda_i$)	SCOV	Distribution	Mean ($1/\mu_i$)	SCOV	
1	Lognormal	0.25	2	Lognormal	5	1.5	20
2	Lognormal	0.5	1.5	Lognormal	10	2	20
3	Exponential	0.25	1	Lognormal	5	1	20
4	Lognormal	0.1	0.8	Lognormal	2	0.8	20
5	Lognormal	0.2	2	Lognormal	3	0.5	15

model (M2) are the approximate condition of the probabilistic constraint (2.4) in the model (M1). Constraints (3.21) and (3.22) are identical to the result in equation (3.13), which denotes the mean and variance of the inter-arrival time of aggregated arrival process into server j. Constraints (3.23) and (3.24) are the results from equation (3.14), which denotes the mean and variance of the workload (job size) of aggregated arrival process into server j. Constraints (3.22) and (3.25) determine the variability parameters to fit the size of uncertainty sets to target probability level $1 - \varepsilon_j$ by the method explained in Section 3.5. Constraint (3.26) bounds the approximate sojourn time quantile as in equation (3.17).

4 Numerical experiments

Here, we show the numerical results of applying the static policy derived through solving the model (M2) in Section 3. As mentioned in Section 1, most of the operation-level dynamic control algorithms for data centers only targeted routing or scaling. A few considering both simultaneously narrowly considered the sojourn time-related probabilistic QoS constraints. Hence, we select the dynamic speed scaling and load balancing algorithm in Ko and Cho (2014) as a benchmark model. For a meaningful comparison, we design our experimental settings similar to those in Ko and Cho (2014). The experimental details, including solving the optimization model (M2) and running a simulation, are explained in Section 5 in the online supplement.

Section 4.1 explains the simulation results of the proposed static policy and the reference dynamic control. Additionally, we compare the power consumption and violation probability by changing the QoS-related parameters, δ and ε. Section 4.2 provides discussions about the proposed static control. Section 4.2.1 discusses the circumstance under which the static policy is relatively appealing and draws insights from the comparison with the dynamic approach. Section 4.2.2 discusses the

Table 4: Server setting

Server (j)	x_j^{min}	x_j^{max}	K_j	α_j	n_j	δ_j	ε_j	$\mathcal{A}_j$
1	5	100	150	1/3	3	δ	ε	1
2	7	102	250	0.2	3	δ	ε	1
3	6	99	220	1	3	δ	ε	1,2
4	5	105	150	2/3	3	δ	ε	1,2,3
5	7	100	300	0.8	3	δ	ε	2,3
6	8	102	350	0.4	3	δ	ε	2,3
7	6	100	220	3/7	3	δ	ε	3
8	7	105	350	0.5	3	δ	ε	4,5
9	8	102	400	0.6	3	δ	ε	4,5
10	10	105	700	4/9	3	δ	ε	5

Power function : $C_j(x_j) = K_j + \alpha_j x_j^{n_j}$, QoS constraint : $\mathbb{P}(S_j \geq \delta_j) \leq \varepsilon_j$

feasibility and scalability of the proposed method.

4.1 *Comparison with dynamic control*

We consider a system with five applications and ten servers as Ko and Cho (2014). For illustrative purposes, we use a small-size problem for comparison and discuss the scalability issue of solving a large-size problem later in Section 4.2.2. The settings for the arrival processes of each application and server are listed in Tables 3 and 4, respectively. SCOV means the squared coefficient of variation.

The arrival processes of the applications are independent renewal processes consisting of inter-arrival times and workloads following general distributions. We assume that the servers' power functions are cubic functions ($C_j(x_j) = K_j + \alpha_j x_j^3$). Each server can have its speed ranges ($[x_j^{min}, x_j^{max}]$), parameters of the power function (K_j, α_j), and a set of applications it can handle ($\mathcal{A}_j$). For simplicity, we use the same $\delta = \delta_j$ and $\varepsilon = \varepsilon_j$ for all $j \in \mathcal{S}$ as in Ko and Cho (2014); note that having different δ_j and ε_j does not affect the computational complexity. The routing probability matrix and the server speeds of the static policy are obtained by solving the optimization model (M2) with an open-source optimization solver IpOpt (version 0.7.0).

Table 5: Comparison result for $\delta = 5$ and $\varepsilon = 0.05$

Control Type	Average Power Consumption (per unit time)	Server j									
		j	1	2	3	4	5	6	7	8	9	10
Dynamic	18421.07	$\mathbb{E}[x_j]$	9.824	13.054	8.220	10.691	9.779	14.499	11.458	13.544	12.372	12.604
		$\mathbb{P}(S_j \geq 5)$	0	0	0.00027	0	0	0.00007	0	0	0	0
Static	16767.49	$\mathbb{E}[x_j]$	13.834	17.324	18.903	17.126	15.313	25.875	15.382	14.297	13.050	15.291
		$\mathbb{P}(S_j \geq 5)$	0.0246	0.0248	0.0075	0.0061	0.0113	0.0098	0.0407	0.0118	0.0118	0.0409

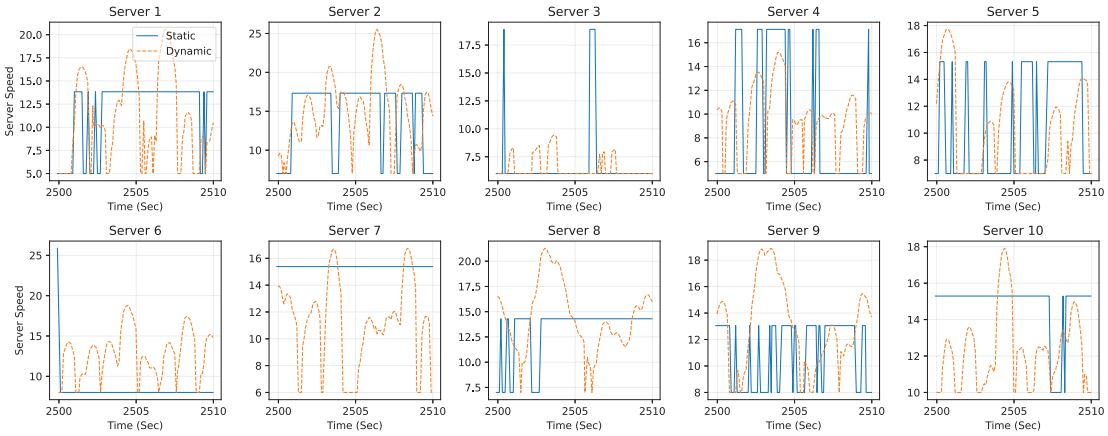


Figure 2: Server speeds for $\delta = 5$ and $\varepsilon = 0.05$

First, we take a close look at a specific operation example under $\delta = 5$ and $\varepsilon = 0.05$, or the QoS constraints being $\mathbb{P}(S_j \geq 5) \leq 0.05,\ \forall j \in \mathcal{S}$. Table 5 summarizes the experimental results. The average power consumption per unit time of the static policy is approximately 9% lower than that of the dynamic algorithm. The dynamic algorithm shows violation probabilities far smaller than the target value $\varepsilon = 0.05$, whereas the proposed static control shows violation probabilities that are not extremely low. It might seem counterintuitive that the mean server speed, $\mathbb{E}[x_j]$, is generally higher under the static control policy, which has lower average power consumption. However, this phenomenon can be explained by considering the characteristics of the control policies and the power function used in the study. The power function is assumed to be cubic, meaning that power consumption increases much more rapidly as server speed increases. Static control operates servers at a constant speed regardless of workload (except for idle time), while dynamic control adjusts the speed of the servers based on the remaining workload. Dynamic control is generally more energy efficient during low-traffic periods. However, when traffic suddenly increases, it might operate at a higher speed for some time to meet QoS constraints, leading to higher energy consumption. Contrastingly, static control will have a manageable backlog of workloads, because it always processes services at a fixed speed, if not idle. As a result, static control might lead to lower average energy consumption, related to the third-order moment of server speed, although it has a higher average operating speed, the first-order moment, compared to dynamic control, particularly in environments with weaker QoS constraints.

Figure 2 shows a part of the server speeds from the simulation result. The dashed and solid

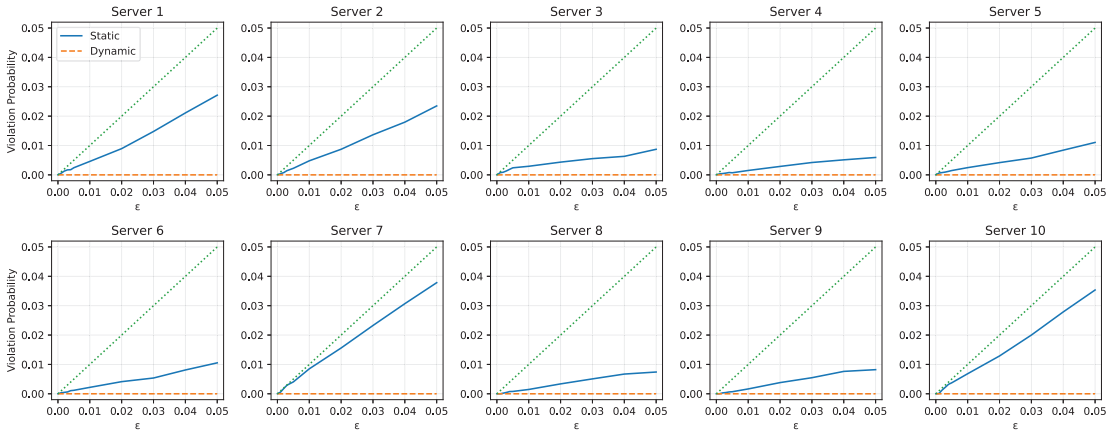


Figure 3: Violation probabilities for $\delta = 5$

lines denote the server speeds from the dynamic and static control over time, respectively. Under the proposed static policy, servers run at the predetermined speed derived from the optimization model only when unfinished jobs are left and run at the lowest possible speed x_j^{min} when the server j is idle. Thus, the server speed under static policy appears like a step-wise function. However, the server speed under dynamic control appears nearly continuous because it changes according to the remaining workload at the moment. In summary, the server speed in the dynamic control continuously varies, whereas the static control speed switches only between two values similar to a bang-bang type control.

Next, we examine the violation probabilities under the dynamic and static policies with different target probabilities ε's and the same $\delta = 5$. The results are plotted in Figure 3. Both policies successfully satisfy QoS conditions; however, the result of the dynamic control is overly conservative, particularly with larger ε, that is, the dashed lines denoting the violation probabilities of the dynamic control fall below the dotted lines denoting the target value ε but significantly close to zero. The solid lines denoting the result of the static policy, however, indicate an adequate level of violation probability while satisfying QoS conditions.

Finally, we compare the energy consumption of both policies as δ and ε vary. Figure 4 shows how the average power consumption changes as ε grows for a specific δ under dynamic (dashed lines) and static (solid lines) policies. As the response time threshold δ or the violation probability bound ε increases, the relative performance of the static policy improves. We expected this result because our construction of a tractable condition that approximates the QoS constraint is tighter

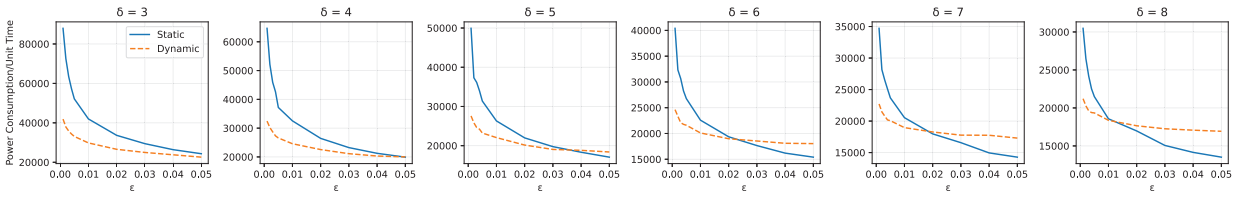


Figure 4: Average power consumptions for $\delta = 3$ to 8

than the heavy-traffic-based approach. For the detailed discussion about the sojourn time quantile estimation, we refer the readers to Section 2 in the online supplement. Thus, we derive a tighter but approximative sojourn time upper bound by using the concepts of the CLT and uncertainty sets. The fundamental difference of the operation policy, whether dynamic or static, causes the dynamic policy to outperform under the strict QoS conditions. However, under the loose QoS conditions, the benefits of our robust queueing theoretic approach appears to overcome the structural limitation of static operation. When $\varepsilon = 0.05$ and $\delta = 8$, the static policy consumes approximately 73% of the energy of the dynamic one while successfully satisfying the QoS conditions.

4.2 *Discussion on static control*

This section assesses the effectiveness of the proposed static control approach from various angles. Section 4.2.1 examines the situations in which the static policy is particularly appealing and provides insights based on a comparison with the dynamic approach. Section 4.2.2 explores the feasibility and scalability of the proposed method for deriving the static policy.

4.2.1 *Discussion on relative performance*

The conservative result in the violation probability described in Section 4.1 originates from how tightly the QoS conditions are satisfied. Mathematically, the aforementioned phenomenon is related to the derivation of the approximate conditions for QoS constraints. Under the strict QoS conditions—that is, small ε and δ values—the traffic intensity may not be heavy enough to apply the heavy-traffic approximations. Furthermore, Ko and Cho (2014) used a loose upper bound of the heavy-traffic result to make the constraint linear, making the result more conservative even under the weak QoS conditions. However, the robust queueing theory does not require assumptions on

traffic intensity to construct uncertainty sets, and the proposed static policy takes advantage of it.

We also investigate the effect of different factors, more than ε and δ, on the relative performance of the static policy. Because of space limitation, we refer the reader to Section 4 in the online supplement for the experimental details, and we only discuss the results here.

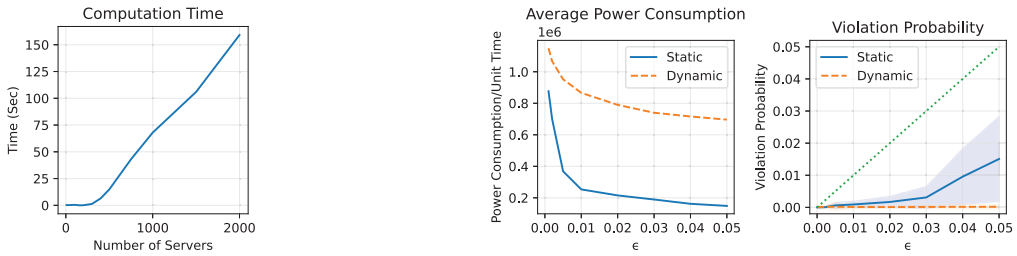
First, the dynamic policy performs relatively better as the arrival rate increases. We can interpret that a higher arrival rate makes traffic intensity closer to one, and consequently, the heavy-traffic approximation of the dynamic control becomes more accurate. Addition, as the variance of the inter-arrival time distribution of the external arrival process increases, the static policy performs relatively better. A larger variance of the inter-arrival time causes a larger variance in the system's status, consequently increasing server speeds to meet the QoS conditions. This variability causes the traffic intensity and accuracy of the heavy-traffic approximation of the dynamic policy to decrease, while the static control can respond resiliently to varying ρ.

Second, as the power functions of servers become more heterogeneous, the static policy performs relatively better. Generally, the performance of the static policy is known to degrade as heterogeneity within a system grows (Chen and Li, 2015). We, however, find that the way of handling the QoS conditions can produce this counter-intuitive result. We observe that service requests are routed to more energy-efficient servers. If the heterogeneity in the power functions increases, this tendency becomes more apparent; there are busy servers and idle servers. The accuracy of the heavy-traffic approximation varies among servers (that is, accurate for servers in heavy traffic and inaccurate with low traffic). The robust queueing theory in the static policy expects consistent accuracy because of its insensitivity to the traffic intensity.

4.2.2 *Additional discussions*

In this section, we discuss the feasibility and scalability of the proposed method. We provide experimental results with our interpretations of them.

For the feasibility of the problem, we successfully obtain the robust control policy under the setting explained in Section 4.1, where ε and δ values are in the ranges $[0.001, 0.05]$ and $[3, 8]$, respectively. In addition, via numerical experiments, we confirm that we could obtain the optimal server speeds and routing probability matrix for a fairly large range of parameters (i.e., ε and δ seem to practically have no restriction). Even for the strict conditions with $\varepsilon = 10^{-5}$ or $\delta = 1$, we



(a) Computation time under different system scales (b) Comparison results for a system with 100 servers

Figure 5: Experiment results for scalability analysis

successfully obtain the optimal solution of (M2).

For the scalability, the dynamic algorithm in Ko and Cho (2014) requires only real-time monitoring of servers and communications between servers and a router. There are practically no scalability issues because it is a distributed algorithm. However, the proposed static policy requires solving the optimization problem (M2). We conduct numerical experiments to confirm whether the proposed methodology is applicable when the system size is large. We measure the computation time to obtain the static policy by solving the optimization problem (M2) with different system sizes. Because of space limitation, we defer the related experimental details to Section 3 in the online supplement and briefly discuss the result here. We successfully obtain the optimal solution of the optimization model and the corresponding static policy, even for a large system scale. Figure 5a shows the computation time according to the number of servers with QoS condition parameters $\delta = 8,\ \varepsilon = 0.05$. The optimization problem for the system with 2,000 servers can be solved within 3 minutes. Figure 5b depicts the average power consumption per unit time and the mean ($\pm$ standard deviation) of violation probabilities of all servers for the system with 100 servers regarding the QoS conditions with $(\delta,\ \varepsilon) = (8,\ 0.001)$. The result shows that the static policy outperforms the dynamic control for almost every ε because of the increased variability of the arrival process. Additionally, we observe that the violation probabilities are not extremely low but still below threshold ε.

5 Conclusion

In this study, we proposed a methodology to derive an energy-efficient static policy while maintaining a satisfactory level of service. Each server was modeled as a $G/G/1/PS$ queue, and QoS

conditions restricted the tail probabilities of response times in the mathematical model. The proposed approach based on the robust queueing theory used uncertainty sets to define the domain of stochastic primitives: inter-arrival times and workloads.

We derived an approximative upper bound of sojourn times from the uncertainty sets and proposed a method to determine the size of the uncertainty sets, that is, variability parameters. Numerical experiments confirmed that the sojourn time bound accurately approximates the probabilistic quantile of steady-state sojourn time. We constructed a tractable inequality that replaces the QoS constraint, which was initially a probabilistic statement. The optimal solution of the reformulated optimization model deduced a static policy consisting of routing probability matrix and server speeds.

The results of the numerical experiments show that under the weak QoS conditions with a generous response time threshold and violation probability level, the proposed static policy outperformed the dynamic algorithm of Ko and Cho (2014) in energy consumption while satisfying the QoS conditions. Moreover, the proposed static policy is practically scalable to relatively large systems.

In future research, the current model could be refined by including the concept of multi-tier architecture and multi-core processors to represent a more realistic system. Additionally, we could consider an alternative design for uncertainty sets in different disciplines (e.g., last come first served and shortest remaining time first). Subsequently, the analysis procedure for various performance measures could also be devised. We could extend the static control policy proposed in this study to consider time-varying arrival processes. A dynamic control algorithm based on the robust queueing theory that utilizes real-time information in constructing uncertainty sets could also be investigated.

Acknowledgements

We would like to thank the Editor, Department Editor, Associate Editor and anonymous reviewers for their insightful comments to strengthen this paper. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1A2C1094699 and NRF-2021R1A4A1031019) and in part by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0008691, HRD Program for Industrial Innovation).

Notes on contributors

Seung Min Baik received the B.S. degree in industrial and management engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2016, where he is currently pursuing the Ph.D. degree in industrial and management engineering. His research interests include robust decision-making in stochastic systems and reliability engineering in stochastic simulation models.

Young Myoung Ko is an associate professor in the Department of Industrial and Management Engineering at Pohang University of Science and Technology (POSTECH), Pohang, South Korea. He received the B.S. and M.S. degrees in industrial engineering from Seoul National University, Seoul, South Korea, and the Ph.D. degree in industrial engineering from Texas A & M University, College Station, TX, USA, in 1998, 2000, and 2011, respectively. His research interests include, but are not limited to, simulation and optimization of stochastic systems, such as telecommunication networks, ICT infrastructure, and renewable energy systems.

References

Amazon (2019) Sustainability: Thinking Big. Available at https://sustainability.aboutamazon.com/2018-sustainability-report-english-uk.pdf.

Andrew, L. L., Wierman, A., and Tang, A. (2009) Optimal speed scaling under arbitrary power functions. *ACM SIGMETRICS Performance Evaluation Review*, **37**(2), 39–41.

Ata, B. and Shneorson, S. (2006) Dynamic control of an M/M/1 service system with adjustable arrival and service rates. *Management Science*, **52**(11), 1778–1791.

Baccarelli, E., Naranjo, P. G. V., Shojafar, M., and Scarpiniti, M. (2017) Q*: Energy and delay-efficient dynamic queue management in TCP/IP virtualized data centers. *Computer Communications*, **102**, 89–106.

Bandi, C., Bertsimas, D., and Youssef, N. (2015) Robust queueing theory. *Operations Research*, **63**(3), 676–700.

Bilal, K., Fayyaz, A., Khan, S. U., and Usman, S. (2015) Power-aware resource allocation in computer clusters using dynamic threshold voltage scaling and dynamic voltage scaling: comparison and analysis. *Cluster Computing*, **18**, 865–888.

Brandt, A. and Brandt, M. (2006) A sample path relation for the sojourn times in G/G/1- PS systems and its applications. *Queueing Systems*, **52**, 281–286.

Chandra, P. K. and Sahoo, B. (2009) Performance analysis of load balancing algorithms for cluster of video on demand servers, in *2009 IEEE International Advance Computing Conference*, IEEE, pp. 408–412.

Chen, L. and Li, N. (2015) On the interaction between load balancing and speed scaling. *IEEE Journal on Selected Areas in Communications*, **33**(12), 2567–2578.

Cho, Y. and Ko, Y. M. (2018) Energy efficiency of data center operating practices: Server clustering, powering on/off, and bang–bang control. *Networks*, **71**(2), 107–119.

Cho, Y. and Ko, Y. M. (2020) Stabilizing the virtual response time in single-server processor sharing queues with slowly time-varying arrival rates. *Annals of Operations Research*, **293**(1), 27–55.

Cho, Y. and Ko, Y. M. (2022) Power-and QoS-aware job assignment with dynamic speed scaling for cloud data center computing. *IEEE Access*, **10**, 38284–38298.

Cisco, C. (2018) Cisco Global Cloud Index: Forecast and Methodology, 2016–2021, Cisco, San Jose. Available at https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_whit

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. (2017) Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms, in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 153–167.

Gallego Arrubla, J. A., Ko, Y. M., Polansky, R. J., Pérez, E., Ntaimo, L., and Gautam, N. (2013) Integrating virtualization, speed scaling, and powering on/off servers in data centers for energy efficiency. *IIE Transactions*, **45**(10), 1114–1136.

Gautam, N. (2012) *Analysis of Queues*. CRC Press, LLC, Boca Raton, Florida, United States

George, J. M. and Harrison, J. M. (2001) Dynamic control of a queue with adjustable service rate. *Operations research*, **49**(5), 720–731.

Google (2019) Google Environmental Report 2019. Available at https://sustainability.google/reports/environmental-report-2019/.

Gul, B., Khan, F. G., and Ahmed, I. (2016) Analyzing virtualization based energy efficiency techniques in cloud data centers. *International Journal of Computer Science and Information Security (IJCSIS)*, **14**(6).

Hölzle, U. (2020) Announcing 'round-the-clock clean energy for cloud, *Google Cloud Blog*.

Horvath, T., Abdelzaher, T., Skadron, K., and Liu, X. (2007) Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, **56**(4), 444–458.

Jiang, C., Han, G., Lin, J., Jia, G., Shi, W., and Wan, J. (2019) Characteristics of co-allocated online services and batch jobs in internet data centers: a case study from Alibaba cloud. *IEEE Access*, **7**, 22495–22508.

Jones, N. *et al.* (2018) How to stop data centres from gobbling up the world's electricity. *Nature*, **561**(7722), 163–166.

Karpowicz, M. P. (2016) Energy-efficient CPU frequency control for the Linux system. *Concurrency and Computation: Practice and Experience*, **28**(2), 420–437.

Kingman, J. F. (1970) Inequalities in the theory of queues. *Journal of the Royal Statistical Society: Series B (Methodological)*, **32**(1), 102–110.

Ko, Y. M. and Cho, Y. (2014) A distributed speed scaling and load balancing algorithm for energy efficient data centers. *Performance Evaluation*, **79**, 120–133.

Lindley, D. V. (1952) The theory of queues with a single server, in *Mathematical Proceedings of the Cambridge Philosophical Society*, Cambridge University Press, Volume **48**, pp. 277–289.

Pelley, S., Meisner, D., Wenisch, T.F., and VanGilder, J.W. (2009) Understanding and abstracting total data center power, in *Workshop on Energy-Efficient Design*, Citeseer, Volume **11**, pp. 1–6.

Rizvandi, N. B. and Zomaya, A. Y. (2012) A primarily survey on energy efficiency in cloud and distributed computing systems. *arXiv preprint arXiv:1210.4690*.

Rong, H., Zhang, H., Xiao, S., Li, C., and Hu, C. (2016) Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, **58**, 674–691.

Scarpiniti, M., Baccarelli, E., Naranjo, P. G. V., and Uncini, A. (2018) Energy performance of heuristics and meta-heuristics for real-time joint resource scaling and consolidation in virtualized networked data centers. *The Journal of Supercomputing*, **74**, 2161–2198.

Uddin, M., Darabidarabkhani, Y., Shah, A., and Memon, J. (2015) Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: A review. *Renewable and Sustainable Energy Reviews*, **51**, 1553–1563.

Urgaonkar, B., Rosenberg, A. L., and Shenoy, P. (2007) Application placement on a cluster of servers. *International Journal of Foundations of Computer Science*, **18**(05), 1023–1041.

Whitt, W. and You, W. (2018) Using robust queueing to expose the impact of dependence in single-server queues. *Operations Research*, **66**(1), 184–199.

Whitt, W. and You, W. (2019) Time-varying robust queueing. *Operations Research*, **67**(6), 1766–1782.

Wierman, A., Andrew, L. L., and Tang, A. (2009) Power-aware speed scaling in processor sharing systems, in *IEEE INFOCOM 2009*, IEEE, pp. 2007–2015.

Wierman, A., Andrew, L. L., and Tang, A. (2012) Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation*, **69**(12), 601–622.

Xiong, Z., Cheng, Y., Cai, L., and Cai, W. (2019) Online power-aware deployment and load distribution optimization for application server clusters. *IEEE Access*, **7**, 91080–91092.

Zhang, J. and Zwart, B. (2008) Steady state approximations of limited processor sharing queues in heavy traffic. *Queueing Systems*, **60**(3-4), 227–246.