Insensitive Job Assignment with Throughput and Energy Criteria for Processor-Sharing Server Farms

Zvi Rosberg, Yu Peng, Jing Fu, Jun Guo, *Member, IEEE,* Eric W. M. Wong, *Senior Member, IEEE,* and Moshe Zukerman, *Fellow, IEEE*

***Abstract*—We study the problem of stochastic job assignment in a server farm comprising multiple processor-sharing servers with various speeds and finite buffer sizes. We consider two types of assignment policies, *without jockeying*, where an arriving job is assigned only once to an available server, and *with jockeying*, where a job may be reassigned at any time. We also require that the underlying Markov process under each policy is *insensitive*. Namely, the stationary distribution of the number of jobs in the system is independent of the job size distribution except for its mean. For the case without jockeying, we derive two insensitive heuristic policies, one aims at maximizing job throughput and the other trades off job throughput for energy efficiency. For the case with jockeying, we formulate the optimal assignment problem as a semi-Markov decision process and derive optimal policies with respect to various optimization criteria. We further derive two simple insensitive heuristic policies with jockeying, one maximizes job throughput and the other aims at maximizing energy efficiency. Numerical examples demonstrate that, under a wide range of system parameters, the latter policy performs very close to the optimal policy. Numerical examples also demonstrate energy/throughput tradeoffs for the various policies and, in the case with jockeying, they show a potential of substantial energy savings relative to a policy that optimizes throughput.**

***Index Terms*—Server farms, energy efficiency, job assignment, insensitivity, processor sharing.**

I. Introduction

Increasing demand for data processing and storage in nearly every sector of the economy has led to a tremendous growth of data centers. Server farms in data centers often consist of thousands of servers and a cooling infrastructure. Energy efficiency of server farms is important considering greenhouse emissions concerns. The U.S. Environmental Protection Agency estimated that servers and data centers consumed about 1.5% of total U.S. electricity consumption in 2006, more than doubled since 2000 [1]. This energy consumption figure along with its increasing growth underscore the need for efficient energy management of data centers in facilitating a low carbon economy in the information age [2].

Part of the overall approach for energy reduction in data centers is to manage the power consumption of server farms. The first step is to design power-aware hardware that can be controlled as to balance between energy consumption and processing speed. With this capability, researchers have proposed speed scaling methods to optimize energy consumption by controlling the server speeds based on their carried load, e.g., [3]–[7]. More recently, researchers have proposed right-sizing techniques to dynamically activate/deactivate servers for energy conservation [8].

The approach that we take in this paper is to investigate how to assign user jobs, with throughput and energy criteria, in server farms as a function of server speeds and the current workload. Our approach provides a way for optimizing energy consumption by controlling carried load on the networked servers. In our model, we consider heterogeneous servers running at fixed speeds. The different server speeds are justified as different servers may be purchased at different times from different vendors using different technologies and designs. For local fine tuning, our approach can also be combined with local speed scaling at each server.

The problem studied here is a stochastic job assignment in a server farm comprising multiple processor-sharing (PS) servers with various speeds and finite buffer sizes. PS server farms are an important architecture [9] and suitable for modeling web-server systems [10]–[14]. Under PS, all existing jobs at each server share the processing capacity and are served at equal rates. PS enables fair processing of jobs, in contrast to the first-come-first-served (FCFS) service discipline. This is desirable in web-server systems where the file size distribution is known to have high variability [15].

An important property of our job assignment policies is known as *insensitivity*. Namely, the stationary (steady-state) distribution of the number of jobs in the system depends on the job size distribution only through its mean. The reasoning for such requirement is to achieve *robustness* and *predictability* of the deployed server farm. Namely, to assure the accuracy and predictability of analytical tools under a wide range of job size distributions. The importance of designing systems that possess the *insensitivity* property was also discussed in [16], [17] and in [18, Ch. 11].

In this paper, we study two types of insensitive assignment policies. One is referred to as *assignment without jockeying*, where a newly arrived job is assigned once to one of the available servers and cannot be reassigned later on. The other is referred to as *assignment with jockeying*, where a job can be reassigned at any time. Assignment without jockeying is less flexible than assignment with jockeying. Computation of the stationary distribution in this class of insensitive policies is difficult because it involves a large multidimensional state space. Assignment with jockeying has more freedom and suits better a single server farm where the servers are collocated in a single physical center and can use e.g. a shared DRAM-storage

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China (email: rosberg@fairflows.com, {yupang2, jingfu6}@student.cityu.edu.hk, {j.guo, eeewong, m.zu}@cityu.edu.hk). The work described in this paper was supported by grants from City University of Hong Kong (Project numbers: 9041794 and 9380044).

[19] or flash-storage [20]. In other cases, the jockeying model can provide performance bounds for server farms. This class of insensitive policies is scalable in computation and hence tractable for assignment optimization.

Most of the job assignment policies proposed in the literature consider servers with infinite buffer sizes and hence aim at optimizing delay performance. We review the related work in Section II. In our study of the job assignment policies, we consider PS servers with finite buffer sizes in situations where a minimum rate is guaranteed to all jobs on a server [16]. Our focus is on throughput and energy criteria. In particular, we aim to control energy consumption of the server farm along with its job throughput. We consider the *energy efficiency* measure of [21], defined in our context as the number of jobs processed divided by the total energy used in the process. Accordingly, given two job assignment policies that consume the same amount of energy, the one that yields a higher job throughput is defined to have a higher energy efficiency.

In Section III, we describe the stochastic model for job assignment to heterogeneous PS servers with finite buffer sizes comprising the PS server farm. Two insensitive assignment policies for the model without jockeying are derived in Section IV, where one aims at maximizing the job throughput and the other trades off job throughput for energy efficiency. In Section V, we characterize the class of insensitive policies with jockeying, and in Section VI, we use the framework of semi-Markov decision process (SMDP) to derive the optimal policy within this class under a family of cost functions. Furthermore, in Section VII, we derive two very simple insensitive heuristic policies with jockeying and show that one maximizes the job throughput and the other aims to maximize the energy efficiency. Numerical results presented in Section VIII illustrate throughput versus energy savings tradeoffs of the various policies with and without jockeying. Finally, conclusions are drawn in Section IX.

II. Related Work

A server farm can be modelled as a queueing system having multiple servers, each with its own queue. Since the work of Haight [22] in 1958, there has been a wealth of studies reported in the literature on job assignment in such a system. The related work can be classified into two broad categories: 1) non-jockeying models where jockeying is not permitted between the parallel queues, and 2) jockeying models where jockeying is allowed. In general, compared to non-jockeying models, a significant improvement of the system performance can be expected with jockeying models.

Join the shortest queue (JSQ) is a classic job assignment policy and has been extensively studied. Under JSQ, an incoming job is assigned to the queue with the least number of existing jobs. Haight [22] derived the stationary distribution under JSQ in a system of two FCFS queues for both non-jockeying and jockeying models, where jockeying is permitted at any time when the two queues differ in length by one. Winston [23] considered a non-jockeying model of multiple FCFS queues with homogeneous servers, and showed that JSQ maximizes the number of jobs served by a certain time. The models of [22], [23] assume Poisson arrival process and exponential job size distribution. Weber [24] extended the result of [23] to the case of an arbitrary arrival process and job size distributions with non-decreasing hazard rate. For non-jockeying models with Poisson arrival process, Whitt [25] showed that the delay performance of JSQ in FCFS systems is sensitive to the variability of the job size distribution. Gupta et al. [11], [12] observed that the delay performance of JSQ in PS systems shows near-insensitivity to the variability of the job size distribution. The non-jockeying policies that we derive in this paper are closely related to JSQ, but they are explicitly designed as insensitive policies for PS server farms with throughput and energy criteria.

Two other well-known job assignment policies are round-robin (RR) and least-work-left (LWL). Under RR, jobs are assigned to the queues in a cyclical fashion. Although RR is simple, JSQ has better performance in both FCFS systems [26] and PS systems [11], [12] because of the advantage of utilizing queue length information in decision making. Under LWL, an incoming job is dispatched to the queue that has the least total outstanding work. LWL is a size-aware policy that requires knowledge of queue length information and the remaining service requirement of each job at each queue. LWL is good for FCFS systems, but its performance in PS systems can be remarkably bad if the variability of the job size distribution is high [9], [11], [12]. Several enhanced size-aware policies have been proposed and analyzed in [13], [27], [28] for PS systems. The policies that we propose in this paper for PS server farms are based on queue length information only, and are explicitly designed as insensitive policies.

Beginning with Haight [22], various jockeying models have been studied in the literature, all of which dealt with JSQ in FCFS systems [29]–[34]. To the best of our knowledge, we are not aware of any study in the literature on jockeying models in PS systems. In this paper, we propose insensitive jockeying policies for job assignment in PS server farms, and our focus is on job throughput and energy efficiency.

For large-scale server farms, researchers have proposed various distributed implementations of job assignment policies, e.g., randomized load balancing [35], join-idle-queue [14], because centralized policies can be unappealing in circumstances where they may require excessive overhead of keeping all up-to-date server states. The policies that we study in the present paper assume knowledge of queue length information and may suit better a single server farm where the servers are collocated in a single physical center.

III. System Model

We model a server farm by K independent servers, each having its own finite buffer for queuing its jobs. For $k = 1, \ldots, K$, denote by B_k the buffer size of server k, and by μ_k its service rate, defined as the units of jobs that can be processed per time unit by the server, e.g., bits per second.

We assume that job sizes (in units) are independent and identically distributed with an absolutely continuous cumulative distribution function (CDF) $F(x)$, $x \geq 0$, and have unity mean without loss of generality. Each server k serves its jobs

at a total speed of μ_k using the PS service discipline. Thus, the service requirement of a job at server k follows a CDF given by $F_k(x) = F(\mu_k \cdot x)$.

We assume that jobs arrive at the system according to a Poisson process with rate λ and are assigned to one of the servers with at least one vacant slot in its buffer. The assignment policy is subject to the control elaborated below. If all buffers are full, the arriving job is lost.

In our design of the job assignment policies, we aim to control the energy efficiency of the server farm along with its job throughput. With current computer server technology, energy consumption of a server is monotonically increasing with the server speed μ and is usually represented by the function

$$\varepsilon(\mu) \propto \mu^{\beta}, \quad \beta > 0 \tag{1}$$

with $\beta = 3$ being the most commonly used value in the literature [4], [36]. However, some researchers argue that $\varepsilon(\mu)$ is not necessarily convex [5], [6]. The optimal policies that we propose in this paper do not assume convexity.

IV. Insensitive Assignment Policies without Jockeying

Our first server farm model considers job assignment policies without jockeying. Within this class of job assignments, we derive two job assignment policies which are *insensitive* to the job size distribution. The first policy, called the *insensitive policy with short queue preference*, or briefly the I-SQP policy, prefers servers with short queue lengths. We argue that I-SQP aims at maximizing the job throughput. The second policy, called the *insensitive policy with short queue and energy preference*, or briefly the I-SQEP policy, modifies I-SQP by trading off job throughput for energy efficiency. These assertions are supported by the optimal policies derived for a model where jockeying is permitted.

The insensitivity property under both policies is implied by the condition derived for a generalized semi-Markov process (GSMP). The model and the condition are briefly presented in the Appendix. In Section IV-A, we derive the insensitivity conditions for a class of assignment policies, where I-SQP and I-SQEP are two particular cases. In Section IV-B and Section IV-C, we derive the stationary distributions under I-SQP and I-SQEP, respectively, and show that both policies induce insensitive GSMPs.

A. Insensitivity conditions for assignment without jockeying

Let $n_k(t)$ be the number of existing jobs in server k at time t and consider the stochastic process $\mathbf{n}(t)$, $t \geq 0$, where $\mathbf{n}(t) = (n_1(t), \ldots, n_K(t))$.

When a new job arrives at time t finding the system at state $\mathbf{n}(t) = \mathbf{n}$, where $\mathbf{n} = (n_1, \ldots, n_K)$, the job is assigned to server k having at least one vacant buffer slot, i.e., $n_k < B_k$, with probability $\gamma_k(\mathbf{n}) > 0$. If all buffers are full, the job is dropped.

The augmented set $\mathcal{S}_0$ (see the Appendix) consists of one element s_0 whose lifetime is exponentially distributed with mean $1/\lambda$. The augmented set $\mathcal{S}_1$ (see the Appendix) consists of K elements $\{s_1, \ldots, s_K\}$ whose lifetimes are distributed according to $F_k(x)$ with means $1/\mu_k$, respectively.

At time 0, the element s_0 is activated and its lifetime decays at a constant rate of λ. When s_0 dies, a new job arrival occurs and the lifetime is instantaneously reborn. Every death follows an instantaneous rebirth measuring the residual time until the next job arrival.

Each arrival assigned to server k activates an instance of the element s_k measuring its lifetime. At every state $\mathbf{n}$, each instance of s_k decays at a state-dependent rate $c(\mathbf{n}, s_k) = \mu_k/n_k$.

Let $\mathbf{e}_k$ denote the unit K dimensional vector with one in the k-th component and zero elsewhere. The transition matrix, $p(\mathbf{n}, s, \mathbf{n}')$, upon a lifetime death is given by

$$p(\mathbf{n}, s, \mathbf{n}') = \begin{cases} \gamma_k(\mathbf{n}), \text{ if } s = s_0, n_k < B_k \text{ and } \mathbf{n}' = \mathbf{n} + \mathbf{e}_k \\ 1, \text{ if } s = s_k, n_k > 0 \text{ and } \mathbf{n}' = \mathbf{n} - \mathbf{e}_k \\ 0, \text{ elsewhere.} \end{cases} \tag{2}$$

Note that the transitions in (2) are conditioned on the event that a respective lifetime component, s, expires. The probabilities that a particular s expires are governed by the independent lifetime distributions and the decaying rates.

It is easily verified that if all $F_k(x)$ are exponentially distributed, the process $\mathbf{n}(t)$ is a Markov process. Denoting its stationary distribution by $\pi(\mathbf{n})$, the partial balance equations are given by

$$\pi(\mathbf{n} + \mathbf{e}_k)\mu_k = \pi(\mathbf{n})\lambda\gamma_k(\mathbf{n}), \ \forall \mathbf{n} : n_k < B_k, \ 1 \leq k \leq K. \tag{3}$$

For every $\mathbf{n} \neq \mathbf{0}$, let $(\mathbf{n}, \mathbf{n} - \mathbf{e}_{k_1}, \mathbf{n} - \mathbf{e}_{k_1} - \mathbf{e}_{k_2}, \ldots, \mathbf{n} - \mathbf{e}_{k_1} - \mathbf{e}_{k_2} - \cdots - \mathbf{e}_{k_{n-1}}, \mathbf{0})$ be a direct path from state $\mathbf{n}$ to state $\mathbf{0}$, which is a loop-free path of length n, where $n = \sum_{k=1}^{K} n_k$. Define the product

$$\Phi(\mathbf{n}) = \gamma_{k_1}(\mathbf{n} - \mathbf{e}_{k_1})\gamma_{k_2}(\mathbf{n} - \mathbf{e}_{k_1} - \mathbf{e}_{k_2}) \cdots \gamma_{k_{n-1}}(\mathbf{n} - \mathbf{e}_{k_1} - \mathbf{e}_{k_2} - \cdots - \mathbf{e}_{k_{n-1}}).$$

Define the distribution

$$\pi(\mathbf{n}) = C\Phi(\mathbf{n}) \prod_{k=1}^{K} \left(\frac{\lambda}{\mu_k}\right)^{n_k} \tag{4}$$

where $C = \pi(\mathbf{0})$ is the normalization constant.

B. Insensitive policy with short queue preference

The I-SQP policy introduces job allocation preference for short queues by setting

$$\gamma_k(\mathbf{n}) = \frac{B_k - n_k}{\sum_{i=1}^{K}(B_i - n_i)}. \tag{5}$$

It is straightforward to verify that (4) with $\gamma_k(\mathbf{n})$ given by (5) satisfies the partial balance equations in (3) for each $\mathbf{n}$, regardless of the selected path used to define $\Phi(\mathbf{n})$. Since partial balancing implies global balancing, (4) with $\gamma_k(\mathbf{n})$ given by (5) is also the stationary distribution under the I-SQP policy and, due to the result of the Appendix, it is insensitive to the job size distribution.

We note that the I-SQP policy is equivalent to an insensitive routing algorithm proposed in [16] for a special example of processor-sharing networks. The blocking probability in this context is obtained as

$$P = \pi(B_1, \ldots, B_K).$$

C. Insensitive policy with short queue and energy preference

The I-SQEP policy is defined by setting

$$\gamma_k(\mathbf{n}) = \frac{B_k - n_k + \left(\frac{\hat{\mu}}{\mu_k}\right)^{\omega} - 1}{\sum_{i=1}^{K}\left[B_i - n_i + \left(\frac{\hat{\mu}}{\mu_i}\right)^{\omega} - 1\right]} \tag{6}$$

where $\hat{\mu} = \max_i \mu_i$. The parameter ω is designed to give preference to slower servers that require less energy consumption. It is clear that the I-SQEP policy reduces to the I-SQP policy when $\omega = 0$. With an increasing ω, there is a higher probability that the I-SQEP policy assigns a job to a slower server that consumes less energy.

As in the case of the I-SQP policy, it is straightforward to verify that (4) with $\gamma_k(\mathbf{n})$ given by (6) satisfies (3). Therefore, the distribution given by (4) with $\gamma_k(\mathbf{n})$ given by (6) is also the stationary distribution under the I-SQEP policy and is insensitive to the job size distribution.

However, it should be noted that the I-SQEP policy is deficient in the sense that it could assign a job to a queue with no vacancy. That is, if server k is busy serving B_k jobs, a new arrival is assigned to it with probability

$$\frac{\left(\frac{\hat{\mu}}{\mu_k}\right)^{\omega} - 1}{\sum_{i=1}^{K}\left[B_i - n_i + \left(\frac{\hat{\mu}}{\mu_i}\right)^{\omega} - 1\right]}$$

in which case it will be lost.

Let $\mathbf{n}_k = (n_1, \ldots, n_K)$ denote the state where $n_k = B_k$ and $n_i \leq B_i$ for all $i \neq k$. Enumerating all $\mathbf{n}_k$ for $k = 1, \ldots, K$, we derive the blocking probability of the I-SQEP policy as

$$P = \sum_{k=1}^{K} \sum_{\mathbf{n}_k} \pi(\mathbf{n}_k)\gamma_k(\mathbf{n}_k).$$

V. Insensitive Assignment Policies with Jockeying

When job jockeying is permitted, jobs can be reassigned to any server with buffer vacancies at any time. It provides significantly more freedom for optimization compared to the case without jockeying. As in Section IV, we constrain ourselves to policies under which the stationary distributions are insensitive.

A. Insensitivity conditions with jockeying

For the assignments with jockeying, we could not use the GSMP framework presented in the Appendix to show insensitivity of the job assignment class we need which is defined below. Nevertheless, we can show the insensitivity of that class by using the symmetric queue model with state dependent service rates derived in [37].

Consider the following class of assignments with jockeying which is sufficiently versatile to tradeoffs between throughput and energy consumption. Let $n = \sum_{k=1}^{K} n_k$, $B = \sum_{k=1}^{K} B_k$. The underlying idea is to define for each policy ϕ a feasible group of server sets $\{\mathcal{T}^{\phi}(n) : \ 1 \leq n \leq B\}$, where $\mathcal{T}^{\phi}(n)$ is the set of servers designated for serving the existing jobs in the system at state n. Since jockeying is allowed, the assignment decisions are made upon each arrival and each departure.

The objective of our job assignment class is to facilitate a tradeoff between job throughput and long run average energy consumption. Note that our policies are functions of the total number of existing jobs and do not consider their residual service requirements.

Since the symmetric queue model with state dependent service rate in [37] is defined for a one dimensional queue length and our system comprises multiple queues, we define for each policy ϕ the following one-to-one mapping Θ^{ϕ} between a two dimensional queue and a one dimensional queue

$$\Theta^{\phi} : \mathcal{Q}_2 \longmapsto \mathcal{Q}_1.$$

The mapping maps the buffer (queue) positions of $\mathcal{Q}_2 \stackrel{\text{def}}{=} \{(k,l) : \ 1 \leq k \leq K, \ 1 \leq l \leq B_k\}$ onto the positions of the *logically combined queue*, $\mathcal{Q}_1 \stackrel{\text{def}}{=} \{1, \ldots, B\}$, so as to match the sets used in the underlying definition of $\{\mathcal{T}^{\phi}(n)\}$.

The mapping Θ^{ϕ} can be defined recursively. This is clarified by the two examples which are given in Section VII; one aims to maximize job throughput and the other aims to maximize energy efficiency. Note that there is more than one mapping Θ^{ϕ} that could match $\{\mathcal{T}^{\phi}(n)\}$. Since each server k uses the PS regime, any mapping that engages the same server sets will do. Additionally, as a result of the insensitivity property of such policies (shown below), the stationary distribution of the underlying stochastic process, $\{n^{\phi}(t), \ t \geq 0\}$, under any such mapping is the same.

To show that for every mapping Θ^{ϕ} (i.e., policy ϕ), the system can be defined as a symmetric queue on the $\mathcal{Q}_1$ domain, we first specify the state dependent service rate over the $\mathcal{Q}_2$ domain.

The total service rate under policy ϕ at state $\mathbf{n}$ is given by

$$\mu^{\phi}(\mathbf{n}) = \sum_{k=1}^{K} \mu_k \cdot \mathcal{I}\{n_k > 0\} = \sum_{k \in \mathcal{T}^{\phi}(n)} \mu_k \stackrel{\text{def}}{=} \mu^{\phi}(n),$$

where $\mathcal{I}\{E\}$ is the indicator function of event E.

Using the PS regime, the proportion of service rate allocated to the job in each position $1 \leq l \leq n_k$ of server k (provided that $n_k > 0$) is given by

$$\gamma^{\phi}(k, l, \mathbf{n}) = \frac{\mu_k}{n_k \mu^{\phi}(\mathbf{n})}.$$

When a job at position l of server k, with its corresponding position in the logically combined queue being $j \stackrel{\text{def}}{=} \Theta^{\phi}(k,l)$, completes its service and leaves the queue, the jobs in positions $j+1, \ldots, n$ of the logically combined queue move to positions $j, \ldots, n-1$, respectively.

A symmetric queue is characterized by the symmetry between service rate allocated to each position and the probability that a newly arrived job will join the system in the corresponding position. Specifically, suppose that a job arrives to the server farm at state $\mathbf{n}$ having a total number of $n < B$

jobs. Then, the total number of jobs is increased to $n+1$, and their new state $\mathbf{n}'$ is uniquely determined by the set $\mathcal{T}^{\phi}(n+1)$ (the set of servers designated for service at state $n+1$). To obtain the required symmetric queue, a new arrival is assigned to position l of server k with a symmetric probability which equals the departure probability from that position, $\gamma^{\phi}(k,l,\mathbf{n}')$. When the new job is placed in its designated position, say (k,l), which corresponds to position $j=\Theta^{\phi}(k,l)$ in the logically combined queue, jobs previously in positions $j,\ldots,n$ of the logically combined queue move to positions $j+1,\ldots,n+1$, respectively.

Using the one-to-one mapping Θ^{ϕ}, all state dependent service rates and position placement probabilities can be expressed on the single dimensional domain, $\mathcal{Q}_1$. With this logically combined queue, each server k serves only the jobs located at its associated positions using a PS regime. To distinguish it from the conventional PS regime where each server equally divides its attention among all waiting jobs, we refer to this version of the PS regime as the *localized PS regime*.

Note that the job departure rate depends on the number of busy servers. Since each server has its own service rate, it follows that for every predetermined position labeling and policy ϕ, the total service rate of the logically combined queue is determined by the queue length, n. Thus,

$$\mu^{\phi}(\mathbf{n})=\mu^{\phi}(n) \quad \text{and} \quad \gamma^{\phi}(k,l,\mathbf{n})=\gamma^{\phi}(\Theta^{\phi}(k,l),n). \quad (7)$$

In [37], it was shown that a stationary symmetric queue in the context of $\mathcal{Q}_1$ is insensitive to phase-type distributions (defined as a mixture of Erlang distributions). It was further established there that a network of nodes with symmetric queues has a stationary distribution that factorizes into a product form over the nodes, and is itself insensitive. The result was extended in [38] to a general distribution.

Due to the insensitivity property, the stationary distribution π^{ϕ} of the process $\{n^{\phi}(t),\ t\geq 0\}$ under any policy with jockeying ϕ, is resolved from the following partial balance equations

$$\pi^{\phi}(n+1)\mu^{\phi}(n+1)=\pi^{\phi}(n)\lambda,\ \forall\ n<B. \quad (8)$$

Denote by $\boldsymbol{\phi}$ the set of all assignment policies with jockeying based on queue length information only. Since $\boldsymbol{\phi}$ does not contain policies which may use specific service requirement realizations, we may consider stationary policies that take decisions only upon arrivals and departures. In the next section we derive the optimal policy over the set $\boldsymbol{\phi}$.

VI. The Long-Run Average Cost Optimal Assignment Policy with Jockeying

The optimal job assignment problem of policies with jockeying can be formulated as an SMDP [39] with a long-run average cost criterion under a variety of cost functions. This assignment problem can be modeled using similar techniques for a general rate control problem of a **single insensitive queue** with Poisson arrivals studied in, e.g., [40], [41]. Numerical methods for computing the optimal service rates in such problems are well known, e.g., *value iteration* and *policy iteration*, based on the *optimality equations* [39, p. 298] with average-cost criterion. Since our problem has a finite state space, the optimal policy can be computed by a simple forward recursive procedure, which is derived next.

Important for the SMDP formulation are the feasible set of servers that can be selected at each state n, $\mathcal{T}(n)$, and its implied individual rate vector, $\boldsymbol{\mu}(n)$. Both determine the total service rate at state n, $\mu(n)$. Note that due to the localized PS regime and the jockeying capability, only the total rate affects the transitions of the underlying queue length, $\{n(t),\ t\geq 0\}$.

For each n, $\mathcal{T}(n)$ can comprise at most $\min(n,K)$ servers whose individual rates are taken from the set $\{\mu_k:\ k=1,\ldots,K\}$. Since each server k can work also at a fractional capacity $0\leq\alpha_k\leq 1$, the set of feasible individual rate vector is

$$\boldsymbol{\mu}(n)=\{\alpha_k\mu_k:\ k\in\mathcal{T}(n)\}$$

and the set of feasible total rates is

$$\mu(n)=\sum_{k\in\mathcal{T}(n)}\alpha_k\mu_k.$$

Thus, each rate $\mu(n)$, which drives the transitions of the process $\{n(t),\ t\geq 0\}$, can be chosen from the compact set

$$\mathcal{A}(n)=\bigcup_{\substack{\mathcal{T}(n)\\ |\mathcal{T}(n)|=\min(n,\,K)}}\ \sum_{k\in\mathcal{T}(n)}\alpha_k\mu_k.$$

Due to the insensitivity property, we may assume exponential service distributions and consider the process $\{n(t)\}$ under an arbitrary assignment policy with jockeying ϕ, where the service rate at each state n is chosen from $\mathcal{A}(n)$. Without loss of generality, we may assume that all server buffers are combined into a single buffer of size B.

Equally important for the computational procedure is the immediate cost function at state n, $\mathrm{C}(n)$. A general function that suits our need is a cost function defined as a sum of the service rates' cost at state n, $\mathrm{R}(n)$, and the holding cost at state n, $\mathrm{H}(n)$. Specifically,

$$\mathrm{C}(n)=\mathrm{R}(n)+\mathrm{H}(n).$$

Since we focus on server energy consumption, which depends not only on the total service rate but also on the individual server rates $\boldsymbol{\mu}(n)$, we generalize the notation of $\mathrm{R}(n)$ into a function $\mathrm{R}(\boldsymbol{\mu}(n))$ which depends on the entire rate vector.

Still, note that the transitions of the SMDP depend only on the total service rate. Moreover, since the state space is finite and each action set $\mathcal{A}(n)$ is compact, the optimal rate at time t is given by a stationary real value rate function $\mu(n)$ of the total rate. That is, the optimal cost cannot be reduced by changing the total service rate between state transitions. We use the notions of the previous sections, ϕ, to denote a stationary policy.

By definition, the optimal policy minimizes the following expected cost of the process $\{n(t),\ t\geq 0\}$ under stationary distribution, i.e., it minimizes over all feasible ϕ

$$V^{\phi}\ \stackrel{\text{def}}{=}\ \sum_{n}\pi^{\phi}(n)\mathrm{C}(n)$$

where π^{ϕ} is the stationary distribution under policy ϕ.

Additionally, the finite state space and the compact action spaces imply the existence of the optimal policy, ϕ^*, and an optimal *value function*

$$V^* = V^{\phi^*} = \min_{\phi} V^{\phi}. \tag{9}$$

The case of energy vs. job throughout

To evaluate the performance of job assignment policies, we next define specific cost functions. One useful service rate cost function is the energy consumed by the server farm when servicing at a total rate of $\mu(n)$. For this purpose, we use the common energy model defined in (1). Note that, although in our examples we use the function $\varepsilon(\mu) = \mu^3$, any positive cost function can be used in our optimal recursive computational procedure.

Since the server farm comprises K servers, i.e., K CPU units, the service rate cost depends on the server constellation used to achieve a total required service rate of μ. Thus, one case of $\mathrm{R}(n)$ accounting the energy consumption at state n is

$$\mathrm{R}(n) \stackrel{\mathrm{def}}{=} \mathcal{E}(\boldsymbol{\mu}(n)) = \sum_{k \in \mathcal{T}(n)} \mu_k^3$$

where $\boldsymbol{\mu}(n) = \{\mu_k : \ k \in \mathcal{T}(n)\}$ is the selected rate vector at state n. That is, $E[\mathcal{E}(\boldsymbol{\mu}(n))]$ is the long-run average energy consumption under stationary conditions when the function $\boldsymbol{\mu}(n)$ is used.

The following two holding cost functions are also useful. For the first function, $\mathrm{H}_1(n)$, let

$$\mathcal{L}(n) = \begin{cases} \lambda, & \text{if } \ n < B \\ 0, & \text{if } \ n = B \end{cases}$$

and, for every $b > 0$, define

$$\mathrm{H}_1(n) \stackrel{\mathrm{def}}{=} -b\mathcal{L}(n).$$

Recalling that λ is the job arrival rate, $E[\mathcal{L}(n)]$ under stationary conditions is the job throughput. Another holding cost function is

$$\mathrm{H}_2(n) \stackrel{\mathrm{def}}{=} \mathcal{Q}(n) = n.$$

That is, $E[\mathcal{Q}(n)]$ is the long-run average queue length under stationary conditions.

Since we are mainly interested to derive a job assignment policy which optimally balances between energy consumption and job throughput, our numerical examples (given below) use the cost function

$$\mathrm{C}(n) = \mathcal{E}(n) + \mathrm{H}_1(n). \tag{10}$$

Now, let

$$V^{\phi}(b) \stackrel{\mathrm{def}}{=} \mathcal{E}^{\phi} - b\mathcal{L}^{\phi}$$

where $\mathcal{L}^{\phi}$ and $\mathcal{E}^{\phi}$ are the job throughput and the average consumed energy, respectively, under stationary conditions given that policy ϕ is employed.

In the case where the SMDP is defined with the cost function of (10), the optimal assignment policy, $\phi^*(b)$, is the solution of

$$\min_{\phi} V^{\phi}(b). \tag{11}$$

That is, $\phi^*(b)$ minimizes the long-run average cost of the system, where the cost rate of the consumed energy is traded off with the job throughput reward at rate b.

A particular interesting job reward rate is b^*, where the energy cost and job throughput reward balances, namely, a b^* that satisfies

$$V^{\phi^*}(b^*) = \mathcal{E}^{\phi^*} - b^*\mathcal{L}^{\phi *} = 0. \tag{12}$$

For this case, we show below that $\phi^*(b^*)$ minimizes the ratio $\mathcal{E}^{\phi}/\mathcal{L}^{\phi}$.

A. An optimal forward recursive procedure

Given that the state space is finite and is given by $\mathcal{S} = \{0, 1, \ldots, B\}$, the minimum average cost of (9) and the optimal rates are derived below by a forward recursive procedure.

For any given b and stationary policy ϕ, let $V_{\phi}^b(n)$ denote the total expected cost until the next visit to state $n = 0$, starting from state n and following policy ϕ. Also, let $V^b(n)$ denote the minimum total expected cost until the next visit to state 0, starting from state $n \geq 0$. For $n = 0$, for the next visit to state $n = 0$ under any policy ϕ, $\{n(t)\}$ must first pass through state $n = 1$, which occurs after $1/\lambda$ on the average. Therefore, we have

$$V_{\phi}^b(0) = \frac{1}{\lambda}\mathrm{C}(0) + V_{\phi}^b(1).$$

Since $\mathcal{A}(0) = \{0\}$, the above relation also holds for $V^b(0)$ and $V^b(1)$.

Before presenting the average-cost optimality equations for the function $V^b(n)$, we need to address the issue that the expected time between state transitions depends on the state and on the policy ϕ. To this end, we employ the *"uniformization" device* [42] and define a virtual exponential clock that triggers events (arrivals and real or virtual service completions) at rate $\lambda + \bar{\mu}$, where $\bar{\mu} = \sum_{k=1}^{K} \mu_k$. The virtual clock defines exponential time transitions occurring at rate $\lambda + \bar{\mu}$. Since for every service policy ϕ, the real service rate at each state n, $\mu(n)$, is not greater than $\bar{\mu}$, each uniform transition is one of the following three types: 1) a new arrival which occurs with probability $\lambda/(\lambda + \bar{\mu})$, 2) a real service completion which occurs with probability $\mu(n)/(\lambda+\bar{\mu})$, and 3) a "dummy" service completion which occurs with probability $(\bar{\mu} - \mu(n))/(\lambda + \bar{\mu})$.

By the *optimality principle of dynamic programming*, the total expected cost function $V^b(n)$ are defined by the following optimality equations:

$$\begin{aligned}(\lambda + \bar{\mu})V^b(n) = \min_{\mu(n) \in \mathcal{A}(n)} \Big\{ & \mathrm{C}(n) + \lambda V^b(\min(n+1, B)) \\ & + \mu(n)V^b(\max(n-1, 0)) \\ & + (\bar{\mu} - \mu(n))V^b(n) \Big\}.\end{aligned} \tag{13}$$

Moreover, the stationary policy that uses the service rate $\mu(n)$ which minimizes the right-hand side of (13) at each state n, is the optimal policy for the total expected cost. If the minimum is not unique, ties are broken by taking the largest minimizer denoted by $\mu^b(n)$.

Rearrange (13), we have

$$(\lambda+\bar{\mu})V^b(n) = \min_{\mu(n)\in\mathcal{A}(n)} \Big\{ \mathrm{C}(n) + \mu(n)V^b(\max(n-1,0)) - \mu(n)V^b(n) \Big\} + \lambda V^b(\min(n+1,B)) + \bar{\mu}V^b(n). \tag{14}$$

Further rearrange (14), we have

$$\lambda\left[V^b(n) - V^b(\min(n+1,B))\right] = \min_{\mu(n)\in\mathcal{A}(n)} \left\{ \mathrm{C}(n) + \mu(n)\left[V^b(\max(n-1,0)) - V^b(n)\right] \right\}. \tag{15}$$

Since the next transition from state $n \geq 1$ must pass through either state $n-1$ or state $\min(n+1,B)$ and the state space is finite, the minimum expected total cost to enter state 0 is finite and can be obtained by solving

$$V^b(n) - V^b(\min(n+1,B)) = \min_{\mu(n)\in\mathcal{A}(n)} \frac{1}{\lambda}\left\{ \mathrm{C}(n) + \mu(n)\left[V^b(\max(n-1,0)) - V^b(n)\right] \right\}$$

which is equivalent to the optimality equations (15) using the differences $V^b(n) - V^b(\min(n+1,B))$. Here, the differences

$$Y^b(n) \stackrel{\text{def}}{=} V^b(n) - V^b(\min(n+1,B)),\ 0 \leq n \leq B-1 \tag{16}$$

are the minimum total expected cost to move from state n to state $n+1$.

For $n=0$, (16) translates into

$$Y^b(0) = \frac{1}{\lambda}\mathrm{C}(0) \tag{17}$$

and, for $1 \leq n < B$, we have

$$Y^b(n) = \min_{\mu(n)\in\mathcal{A}(n)} \frac{1}{\lambda}\Big\{ \mathrm{C}(n) + \mu(n)Y^b(n-1) \Big\}. \tag{18}$$

Equations (17) and (18) comprise a set of forward recursive equations for computing $Y^b(n)$, $1 \leq n < B$, along with the average-cost optimal rates $\mu^b(n)$. The service rate function $\mu^b(n)$ is the optimal average-cost service rate function if and only if $b = b^*$, where b^* satisfies (12).

Let $\tau_\phi(n)$ denote the total time until the next visit to state $n=0$, starting from state n and following policy ϕ. Let ϕ^* denote the optimal policy as a result of the above forward recursive equations. The following respective relations hold for every ϕ and b:

$$V^b_{\phi^*}(0) \begin{cases} > & 0 \\ = & 0 \\ < & 0 \end{cases}$$

if and only if

$$T(b) \stackrel{\text{def}}{=} \frac{V^b_{\phi^*}(0)}{\tau_{\phi^*}(0)} \begin{cases} > & 0 \\ = & 0 \\ < & 0. \end{cases}$$

Also observe that $V^b_{\phi^*}(0)$ is decreasing in b. By starting with two initial costs b_1 and b_2, such that $b_1 \leq b^* \leq b_2$, one can search for the optimal b^* and ϕ^* using an efficient procedure as suggested in [43]. Next, we express the derivation above into a complete procedure.

Forward recursive procedure

1) Start with initial b_1 and b_2 such that $b_1 \leq b^* \leq b_2$ (see below).
2) Set $b = (b_1+b_2)/2$ and use equations (17) and (18) to compute $Y^b(n)$ and $\mu^b(n)$ for every $1 \leq n \leq B$.
3) Employ the insensitivity and use the balance equations of (8) to compute the stationary expected value of $\mathrm{C}(n)$ using the stationary policy ϕ^b induced by the service rates $\mu^b(n)$, $1 \leq n \leq B$. That is, compute the optimal average cost
$$T(b) = \sum_{n=0}^{B} \pi^b(n)\mathrm{C}(n) \tag{19}$$
where
$$\pi^b(n) = \pi^b(0)\prod_{i=1}^{n}\frac{\lambda}{\mu^b(i)},\quad 0 \leq n \leq B.$$
4) Given one of the outcomes in (19) above, continue as follows:
   a) If $T(b) = 0$, then $\phi^*(b) = \phi^*(b^*)$; stop.
   b) If $T(b) > 0$, then the decreasing monotonicity implies $b < b^*$; set $b \to b_1$.
   c) If $T(b) < 0$, then the decreasing monotonicity implies $b > b^*$; set $b \to b_2$.
5) If $b_2 - b_1 < \epsilon$, where ϵ is a predetermined error margin, use $\phi^*(b)$ as an estimator of $\phi^*(b^*)$ and stop; otherwise, go to step 2.

The initial b_1 can be set to 0 here. The initial b_2 can be derived from any stationary feasible policy ϕ, e.g., those introduced in Section VII-A and Section VII-B. That is,

$$b_2 = \frac{\sum_{n=0}^{B}\pi^\phi(n)\mathrm{R}(n)}{\sum_{n=0}^{B}\pi^\phi(n)\mathcal{L}(n)}$$

where

$$\pi^\phi(n) = \pi^\phi(0)\prod_{i=1}^{n}\frac{\lambda}{\mu^\phi(i)},\quad 0 \leq n \leq B.$$

B. The optimality of $\phi^*(b^*)$

In our formulation, the measure of energy efficiency is given by $\mathcal{L}^\phi/\mathcal{E}^\phi$. In the following theorem, we show that $\phi^*(b^*)$ minimizes $\mathcal{E}^\phi/\mathcal{L}^\phi$ and therefore maximizes energy efficiency.

Theorem 1: The job assignment policy with jockeying, $\phi^*(b^*)$, defined by (11) and (12) is optimal, i.e.,

$$\frac{\mathcal{E}^{\phi^*(b^*)}}{\mathcal{L}^{\phi^*(b^*)}} \leq \frac{\mathcal{E}^\phi}{\mathcal{L}^\phi},\quad \forall\ \phi \in \boldsymbol{\phi}.$$

Proof: As shown above, for any given $b > 0$, $\phi^*(b)$ minimizes $V^\phi(b) = \mathcal{E}^\phi - b\mathcal{L}^\phi$ over all $\phi \in \boldsymbol{\phi}$. That is,

$$\mathcal{E}^{\phi^*(b)} - b\mathcal{L}^{\phi^*(b)} \leq \mathcal{E}^\phi - b\mathcal{L}^\phi,\quad \forall\ \phi \in \boldsymbol{\phi}.$$

Particularly, setting $b = b^*$ implies that

$$0 = \mathcal{E}^{\phi^*(b^*)} - b^*\mathcal{L}^{\phi^*(b^*)} \leq \mathcal{E}^\phi - b^*\mathcal{L}^\phi,\quad \forall\ \phi \in \boldsymbol{\phi}. \tag{20}$$

From the left-hand side equality of (20) it follows that

$$b^* = \frac{\mathcal{E}^{\phi^*(b^*)}}{\mathcal{L}^{\phi^*(b^*)}} \tag{21}$$

and from the right-hand side inequality it follows that

$$\mathcal{E}^{\phi} - b^* \mathcal{L}^{\phi} \geq 0. \tag{22}$$

By combining (21) and (22) we have

$$\frac{\mathcal{E}^{\phi}}{\mathcal{L}^{\phi}} \geq b^* = \frac{\mathcal{E}^{\phi^*(b^*)}}{\mathcal{L}^{\phi^*(b^*)}}.$$

■

We will denote the optimal policy $\phi^*(b^*)$ by I-J-OPT, to emphasize that it is an insensitive policy with jockeying. Since it is not given in an explicit form and may require a complex computation, the derivation of heuristic policies has merit.

VII. Heuristic Policies with Jockeying

In this section, we derive two heuristic policies with jockeying. The first policy, called the *insensitive policy with jockeying and with fastest idle server first*, or briefly the I-J-FISF policy, is optimal with respect to the throughput. The second policy, called the *insensitive policy with jockeying and with slowest server first*, or briefly the I-J-SSF policy, approximates the (optimal) policy I-J-OPT and aims at maximizing the energy efficiency.

A. Insensitive policy with jockeying and with fastest idle server first

Without loss of generality, we label the servers in a decreasing order of their speed, i.e., $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_K$. That is, server 1 is the fastest server and server K is the slowest.

With localized PS, the job departure process depends only on the set of busy servers. Thus, for every $n = \sum_{k=1}^{K} n_k$, the instantaneous departure rate from the server farm at state n is maximized if the set of busy servers is set to

$$\mathcal{T}(n) = \{1, \ldots, \min(n, K)\}. \tag{23}$$

This group of sets defines the I-J-FISF policy, denoted by ϕ_1. Note that the I-J-FISF policy satisfies two conditions at any point in time, i.e., 1) no server is idle if a slower server is busy, and 2) no server has more than one job if another server is idle. Since I-J-FISF is a policy with jockeying, by definition, a rearrangement of the existing jobs is always required whenever it becomes necessary to satisfy the two conditions of the policy. One example is when a faster server becomes empty and a slower server still works. Another example is when a slower server becomes empty and a faster server still has more than one job.

The position mapping of ϕ_1, Θ^{ϕ_1}, is defined iteratively as follows. In the first iteration, the first buffer positions of servers $1, 2, \ldots, K$ are mapped to the first K positions of the logically combined queue in the order of the server labels $1, 2, \ldots, K$, inheriting their original server rates. In every subsequent iteration, the next remaining position levels from the remaining buffers, say $m \leq K$ positions, are mapped to the next m positions of the logically combined queue in the order of the server labels. The iterations terminate when all the positions of all server buffers have been mapped.

Position mapping Θ^{ϕ_1} matches the sets $\mathcal{T}(n)$ defined by (23). It also defines a particular assignment with jockeying policy which induces an insensitive policy when new jobs join queue positions according to the probabilities defined by (7), where ϕ is set to ϕ_1.

Under the I-J-FISF policy, we have

$$\mu^{\phi_1}(n) = \sum_{k=1}^{\min(n,K)} \mu_k.$$

It is straightforward to verify that the partial balance equations of (8) are satisfied by the following stationary distribution

$$\pi^{\phi_1}(n) = \begin{cases} \pi^{\phi_1}(0) \prod_{i=1}^{n} \frac{\lambda}{\mu^{\phi_1}(i)}, & \text{if } 0 \leq n \leq K, \\ \pi^{\phi_1}(0) \left(\frac{\lambda}{\mu^{\phi_1}(K)}\right)^{n-K} \prod_{i=1}^{K} \frac{\lambda}{\mu^{\phi_1}(i)}, & \text{if } K < n \leq B, \end{cases} \tag{24}$$

where $\pi^{\phi_1}(0)$ is the normalization constant.

Next, we show that I-J-FISF is optimal in the following sense. Given a policy $\phi \in \boldsymbol{\phi}$, let U_T^{ϕ} be the total units of jobs processed by all servers until time horizon T under policy ϕ, i.e.,

$$U_T^{\phi} = \int_{t=0}^{T} \mu\big(|\mathbf{n}^{\phi}(t)|\big) dt,$$

where $\mathbf{n}^{\phi}(t) = \big(n_1^{\phi}(t), \ldots, n_K^{\phi}(t)\big)$ is the queue length vector at time t under policy ϕ and $|\mathbf{n}^{\phi}(t)| = \sum_{k=1}^{K} n_k^{\phi}(t)$.

Theorem 2: For every $\phi \in \boldsymbol{\phi}$, $\quad \mu_T^{\phi_1} \geq \mu_T^{\phi}, \ \forall\, T$.

Proof: Recall that $\boldsymbol{\phi}$, by definition, is the set of all assignment policies with jockeying based on queue length information only. First note that under policy I-J-FISF, at any time t, the number of busy servers is maximized and for each k, $n_k(t) > 0$ only if $n_i(t) > 0$, for every $i < k$. Next, we prove that a policy which does not satisfy this property, denoted by $(*)$, can be strictly improved for some finite time horizon (using a non-stationary policy).

Suppose that ϕ does not satisfy $(*)$ and we show that it can be strictly improved. Under policy ϕ, there is a first time t_0, where $n_k(t_0) > 0$ and $n_i(t_0) = 0$ for some $i < k$. Consequently, we can improve policy ϕ by a policy $\tilde{\phi}$ which mimics ϕ actions until time t_0. At time t_0, it moves a job, denoted by r^*, from queue k to queue i and waits until the next state transition (arrival or departure) at some time $t_1 > t_0$. At that moment (just before a decision is taken by ϕ), it moves job r^* (if not left already) back to its original queue i. From t_1 onwards, $\tilde{\phi}$ continues to mimic ϕ, with the possible exception in the scenario where r^* has left at time t_1. In this case, r^* is replaced by a dummy job with the same residual service requirement.

Since $\mu_i > \mu_k$ and $t_1 - t_0 > 0$, it follows that

$$U_t^{\tilde{\phi}} > U_t^{\phi}, \ t \in [t_0, t_1)$$

with probability one.

Also, since any policy $\phi \neq$ I-J-FISF can be improved, I-J-FISF is optimal for any finite horizon T. To show that ϕ_1

cannot be improved also with respect to the long run average of the delivered service rate, i.e., job throughput, the argument above shows the following. Any policy which violates property $(*)$ above can be strictly improved unless it violates property $(*)$ on a set of states whose stationary probability is zero. ■

The following corollary, implied from Theorem 2, shows that I-J-FISF is also optimal with respect to job throughput and blocking probability. Let N_T^ϕ be the total number of jobs that have departed from all servers until time horizon T under policy ϕ. Since $\phi \in \boldsymbol{\phi}$ is ergodic, we have

$$\frac{U_T^\phi/T}{N_T^\phi/T} \longrightarrow E(X) \text{ as } T \to \infty \tag{25}$$

where $E(X)$ is the expected job size.

Consider the limits of the departure rates measured in units of jobs and the ones measured in number of jobs, respectively. We have

$$\begin{aligned} \bar{\mu}^\phi &\stackrel{\text{def}}{=} \lim_{T\to\infty} \tfrac{1}{T} U_T^\phi, \\ \mathcal{L}^\phi &\stackrel{\text{def}}{=} \lim_{T\to\infty} \tfrac{1}{T} N_T^\phi. \end{aligned} \tag{26}$$

By definition, $\mathcal{L}^\phi$ is the job throughput, and by (25) and (26), the ratio $\bar{\mu}^\phi/\mathcal{L}^\phi$ is independent of the policy. Also, since the job blocking probability, P^ϕ, is given by $1 - \mathcal{L}^\phi/\lambda$, Theorem 2 implies the following corollary.

Corollary 1: Policy I-J-FISF is also optimal with respect to job throughput and job blocking probability. That is, for every $\phi \in \boldsymbol{\phi}$,

$$\begin{aligned} \mathcal{L}^{\phi_1} &\geq \mathcal{L}^\phi, \\ P^{\phi_1} &\leq P^\phi. \end{aligned}$$

B. Insensitive policy with jockeying and with slowest server first

For notational clarity and differently from Section VII-A, we label the servers according to their energy efficiency, using the convention of $k < k'$ if and only if $\varepsilon(\mu_k)/\mu_k < \varepsilon(\mu_{k'})/\mu_{k'}$. Note that for $\varepsilon(\mu) = \mu^\beta$ and $\beta > 1$, $k < k'$ if and only if $\mu_k < \mu_{k'}$. Thus the server labels for the energy problem are reversed compared to the labels used for the throughput optimization. That is, the smaller the server rate the smaller its label is; i.e., the lower its energy consumption is.

Assuming that energy consumption is proportional to the server operational time, the least energy consumption assignment policy in a non-blocking system (i.e., with unlimited buffer sizes) is attained by maximizing the idle times of the most energy consuming servers at time t. Intuitively, it is attained by shifting the jobs toward the most energy conserving servers. That is, after any change of state $\mathbf{n}$, the jobs are shifted toward the most energy conserving servers (filling up the buffers of the servers with smallest labels). This least energy consumption assignment policy is instrumental for our blocking system. The assignment policy I-J-SSF is defined by specifying the set of servers designated for serving the jobs, at each state $1 \leq n \leq B$, as

$$\mathcal{T}(n) = \{1, \ldots, j\} \tag{27}$$

where $j \leq K$ is the smallest integer satisfying $n \leq \sum_{i=1}^{j} B_i$.

The server sets in (27) indeed define the I-J-SSF policy described above, in which the available servers with lowest rates are designated for service for each state n. Policy I-J-SSF is also denoted by ϕ_2.

The position mapping Θ^{ϕ_2} of ϕ_2 is defined iteratively as follows. The B_1 positions of server 1 (the most energy efficient server) are mapped to the first B_1 positions of the logically combined queue and are associated with the rate and energy parameters of server 1. The B_2 positions of server 2 (the second most energy efficient) are mapped to the following B_2 positions of the logically combined queue and are associated with the rate and energy parameters of server 2. This procedure continues until positions of server K have been mapped.

Position mapping Θ^{ϕ_2} matches the sets $\mathcal{T}(n)$ defined by (27). It also defines a particular assignment and jockeying policy which induces an insensitive policy when new jobs join each queue position with a symmetrical departure probability as defined by (7) where ϕ is set to ϕ_2.

Under the I-J-SSF policy, ϕ_2, we have

$$\mu^{\phi_2}(n) = \sum_{k \in \mathcal{T}(n)} \mu_k$$

and it is straightforward to verify that the partial balance equations of (8) are satisfied by the following stationary distribution

$$\pi^{\phi_2}(n) = \pi^{\phi_2}(0) \prod_{i=1}^{n} \frac{\lambda}{\mu^{\phi_2}(i)}, \quad 0 \leq n \leq B, \tag{28}$$

where $\pi^{\phi_2}(0)$ is the normalization constant.

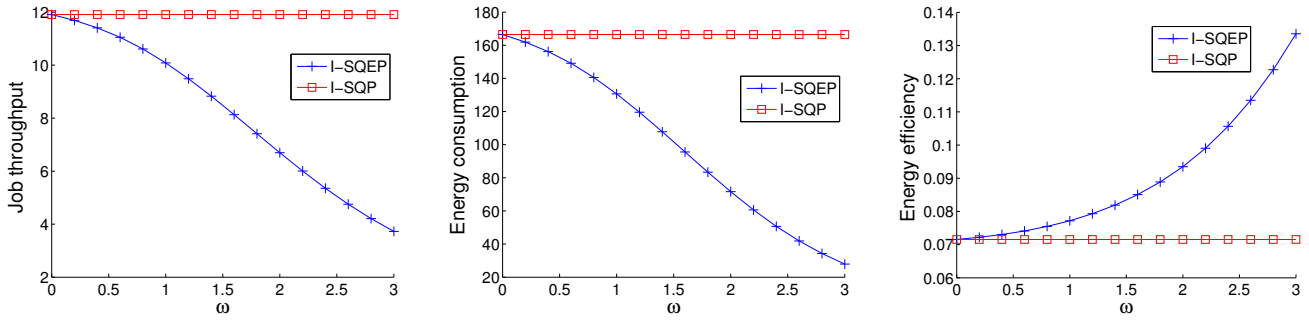
Note that for each k, $\mu^{\phi_2}(n)$ are fixed for each $n \in \{n : \sum_{i=1}^{k-1} B_i < n \leq \sum_{i=1}^{k} B_i\}$.

VIII. Numerical Results

This section provides extensive numerical results that demonstrate the effectiveness of the various policies proposed in this paper and their throughput versus energy savings tradeoffs.

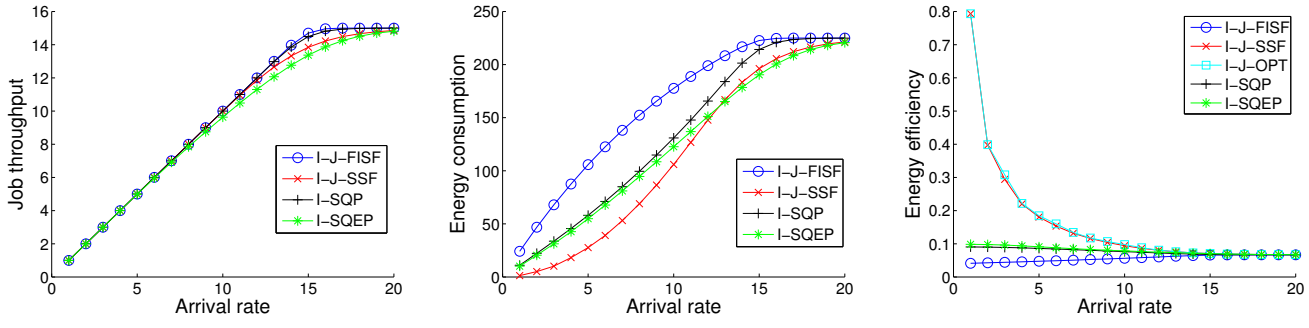
Figure 1 illustrates the performance of the I-SQEP policy under a wide range of values of the parameter ω with the I-SQP policy serving as the benchmark. In this set of experiments, we consider a system with five servers, and set the speed $\mu_k = k$ and the buffer size $B_k = 10$ for each server k. The arrival rate is $\lambda = 12$. Recall that, with a higher value of ω, I-SQEP assigns a job with a higher probability to a slower server that consumes less energy. However, in this way, there is also a higher probability that the job is assigned to a queue with no vacancy and, therefore, the blocking probability of the system is higher. The results in Fig. 1(a) and Fig. 1(b) confirm that, as ω increases, both the job throughput and the energy consumption of I-SQEP decrease. We observe in Fig. 1(c) that the energy efficiency of I-SQEP monotonically increases as ω increases. As expected, I-SQEP reduces to I-SQP when $\omega = 0$. Thus, I-SQEP can effectively trade off job throughput for energy efficiency by tuning the parameter ω.

In Fig. 2 and Fig. 3, we compare the performance of the various policies (with and without jockeying) under various system parameters. Since the number of states in the class



(a) (b) (c)

Fig. 1. Performance of the I-SQEP policy with respect to the parameter ω. (a) Job throughput. (b) Energy consumption. (c) Energy efficiency. The I-SQP policy serves as the benchmark.



(a) (b) (c)

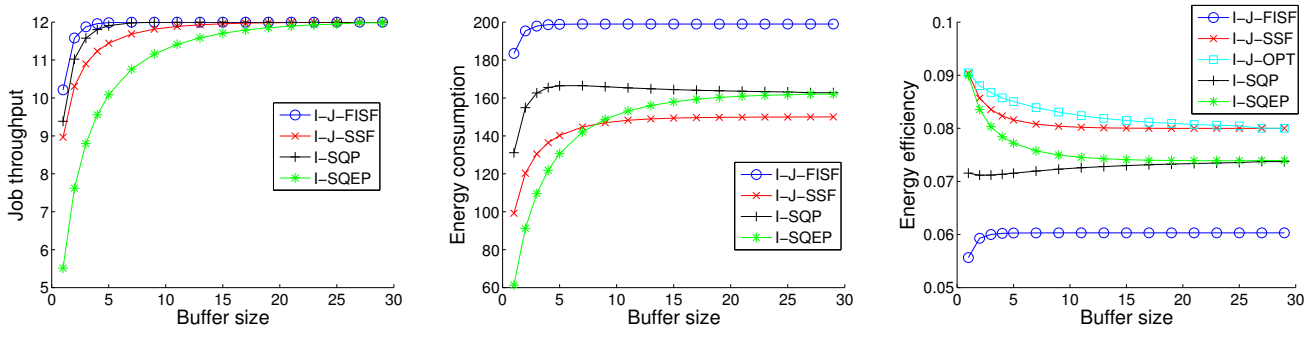
Fig. 2. Performance comparison of various policies in small systems with respect to the arrival rate. (a) Job throughput. (b) Energy consumption. (c) Energy efficiency.

of insensitive policies without jockeying is $\prod_{k=1}^{K}(1+B_k)$, computation of the stationary distribution under I-SQP and I-SQEP is difficult and is limited to cases of small systems. In this set of experiments, we set the parameter $\omega = 1$ for I-SQEP. We consider a system with five servers; the speed of each server k is $\mu_k = k$. The results in Fig. 2 are obtained with the buffer size $B_k = 10$ for each server k and the arrival rate λ varied from 1 to 20. For the results in Fig. 3, the arrival rate is set at $\lambda = 12$ and all servers have the same buffer size which is varied from 1 to 29.

We observe in Fig. 2 that, when the arrival rate is small, all policies achieve almost the same job throughput. However, since the I-J-SSF policy is designed to make more use of the energy conserving servers, it consumes significantly less energy compared to I-SQP, I-SQEP and I-J-FISF. As a result, I-J-SSF yields significantly higher energy efficiency and performs over eight times better than the other policies, especially the throughput-optimal I-J-FISF policy. On the other hand, it is clear that, when the arrival rate increases, the servers' utilization also increases, and it is more likely that all the servers are busy all the time regardless of the policy. Accordingly, all policies converge to the same level of system performance. In Fig. 3, we observe that both the job throughput and the energy consumption increase because more jobs can be admitted as the buffer size increases. Under the various policies, the system performance converges to a certain level at a point where the buffer size is sufficiently large, so that almost all jobs can be admitted and the blocking probability becomes negligible.

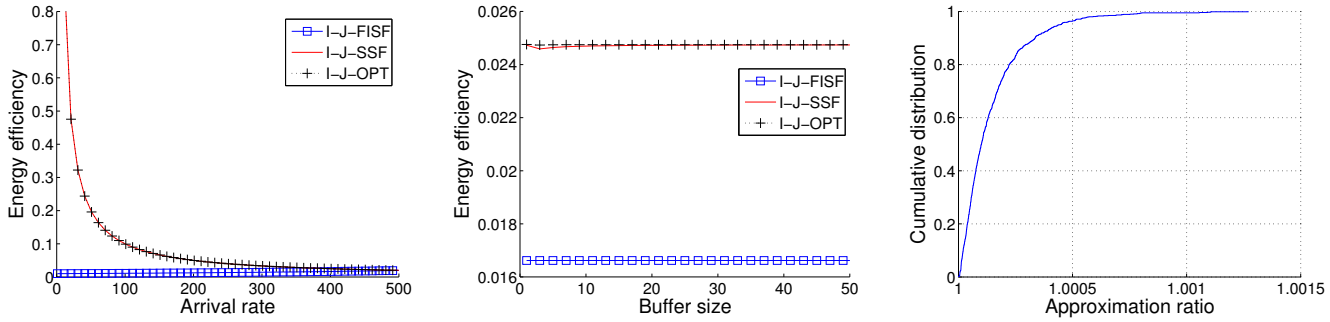
Among the various policies, as shown in Fig. 2 and Fig. 3, **the throughput-optimal I-J-FISF policy always yields the highest job throughput**, but consumes the largest amount of energy and is the least energy efficient. In all cases, we observe that **the (optimal) I-J-OPT policy is indeed the best with respect to the energy efficiency**, and **the energy efficiency of I-J-SSF is very close to that of I-J-OPT**, especially in Fig. 2(c). Although it is observed in Fig. 3(b) that I-SQEP consumes the smallest amount of energy when the buffer size is small (i.e., $1, \ldots, 6$), it is achieved at the expense of sacrificing the job throughput.

Since the class of insensitive policies with jockeying is computationally scalable, we can further examine the effectiveness of I-J-FISF, I-J-SSF and I-J-OPT using large system examples. The energy efficiency results in Fig. 4 are obtained from extensive experiments under a wide range of system parameters for a system comprising 100 heterogeneous servers. In particular,



(a) (b) (c)

Fig. 3. Performance comparison of various policies in small systems with respect to the buffer size. (a) Job throughput. (b) Energy consumption. (c) Energy efficiency.



(a) (b) (c)

Fig. 4. Energy efficiency comparison of policies with jockeying in large systems. (a) With respect to the arrival rate. (b) With respect to the buffer size. (c) Cumulative distribution of the approximation ratio of the I-J-SSF policy to the I-J-OPT policy under various configurations of server speed.

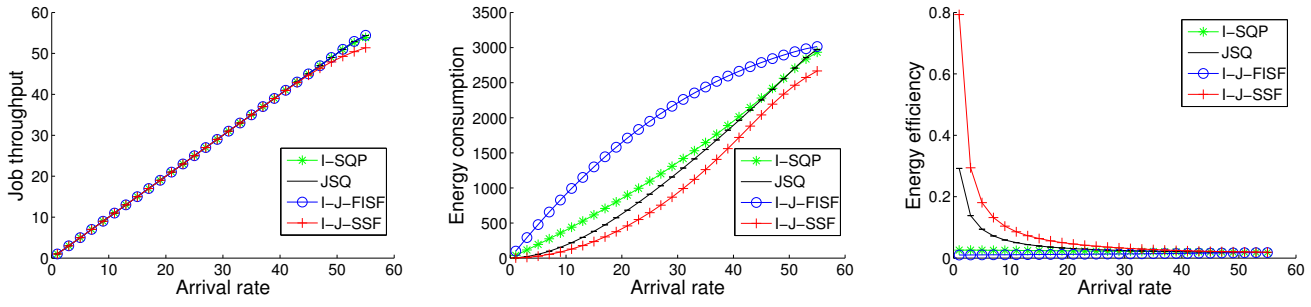
we consider different choices of arrival rate, different choices of buffer size, and various configurations of server speed.

In both Fig. 4(a) and Fig. 4(b), the speed of server k is fixed at $\mu_k = 0.1k$. In Fig. 4(a), we set the buffer size $B_k = 100$ for each server k and vary the arrival rate λ from 1 to 500. In Fig. 4(b), the arrival rate is set at $\lambda = 404$, and all servers have the same buffer size which is varied from 1 to 49. In both figures, we have similar observations of the three policies to those found in Fig. 2(c) and Fig. 3(c) for small system examples.

In Fig. 4(c), for each server k, we set the buffer size $B_k = 100$ and randomly (uniformly) choose the speed of server k from the range $[0.1, 10]$. For each such random configuration of server speed, we set the arrival rate at $\lambda = 0.8 \sum_k \mu_k$. We compare I-J-SSF and I-J-OPT in terms of the *approximation ratio*, defined in this context as the ratio of the energy efficiency of I-J-SSF to that of I-J-OPT. Results are obtained from 1000 experiments, each with a random configuration of server speed, and are plotted in Fig. 4(c) in the form of cumulative distribution. We again observe that **the very simple heuristic I-J-SSF policy performs very close to I-J-OPT** in the scale of that figure. In all the 1000 random experiments, the approximation ratio is smaller than 1.0015. Up to 95% of the observations are smaller than 1.0004.

Figure 5 illustrates the price of insensitivity of the I-SQP policy by comparing it to the JSQ policy. Recall that JSQ always assigns an incoming job to the shortest queue, while I-SQP assigns the job to a shorter queue with a higher "preference" probability to achieve insensitivity. Since there is no analytical solution of JSQ available in the literature, we evaluate its performance by simulation, and we use the deterministic job size distribution, for which JSQ is known to achieve the best performance [11], [12]. All simulation results of JSQ are obtained in the form of an observed mean from multiple independent runs and its confidence interval at the 95% level based on the Student's t-distribution. In this set of experiments, we consider a system with ten servers. For each server k, its speed is $\mu_k = k$, and its buffer size is $B_k = 10$. We vary the arrival rate λ from 1 to 55. We observe in Fig. 5 that, when the arrival rate is small, the throughput is almost the same for the two policies. However, the energy efficiency of I-SQP is significantly worse than that of JSQ. In fact, one can expect that, when the buffer size is large and the arrival rate is small, the preference probabilities of I-SQP are more or less the same. In such cases, even the throughput of I-SQP can be far worse than that of JSQ.

The price of insensitivity of the I-SQP policy can be



(a) (b) (c)

Fig. 5. Performance comparison of various policies with respect to the arrival rate. (a) Job throughput. (b) Energy consumption. (c) Energy efficiency. The results of the JSQ policy are obtained by simulation using a deterministic job size distribution.

mitigated by introducing jockeying to job assignment. As shown in Fig. 5, compared to JSQ, the two insensitive policies with jockeying achieve better performance in their respective objectives, because they have the flexibility to choose the best server at any moment. Although various jockeying models of JSQ have been studied in the literature, they were not designed to yield insensitive policies, which are the focus of the present paper.

IX. Conclusions

We have studied job assignment policies in the popular server farm model comprising multiple heterogenous PS servers with finite buffers. Since server farms are known for their massive energy consumption, we focused on management policies aiming at energy conservation.

Unlike previous studies of management policies, where server speeds are scaled as a function of their workload, or servers are dynamically activated/deactivated for energy conservation, the management policies considered in this study are job assignments to multiple heterogeneous servers. We believe that combining our approach with the server scalability approach would enhance the energy management of server farms. Additionally and differently from other studies, we have designed our assignment policies to be *insensitive*, namely, their stationary distributions depend on the job size distributions only through their means. We argue that insensitivity is a very important aspect of performance evaluation and system deployment, since it enhances commercial systems with the properties of robustness and performance predictability.

We have considered two insensitive policies without jockeying and two insensitive policies with jockeying. We have provided extensive numerical results that demonstrate the effectiveness of the various policies and their throughput versus energy savings tradeoffs. Our main observations are summarized below:

- The two insensitive policies without jockeying, i.e., I-SQP and I-SQEP, do not achieve their objectives as well as their "with jockeying" counterparts because they do not have the flexibility to choose the best server at any moment, and because they involve probabilistic server assignments to achieve insensitivity.
- Compared to the classic JSQ policy, there is a price of insensitivity of the I-SQP policy. However, this price of insensitivity of the I-SQP policy can be mitigated by introducing jockeying to job assignment.
- The performance of the two insensitive policies with jockeying is remarkable in their respective objectives. I-J-FISF is throughput-optimal. I-J-SSF is very close to optimal in the various cases with respect to energy efficiency.

Appendix

A standard framework for proving the insensitivity property of a stochastic process is the generalized semi-Markov process (GSMP) originally derived in [44] and further extended in various papers, e.g., [45], [46] and [47].

We follow the definitions of [46] and the extension of [47] for GSMP with speeds. Let $\mathbf{n}(t)$, $t \geq 0$, be a vector stochastic process on a countable state space $\mathcal{N}$ and $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$ a union of two countable set of indices of possible events that can occur. Each set $\mathcal{S}_i$, $i = 0, 1$, represents different events defined below. To each state $\mathbf{n} \in \mathcal{N}$ corresponds a non-empty finite subset of events $\mathcal{S}(\mathbf{n}) \subset \mathcal{S}$. Each event $s \in \mathcal{S}(\mathbf{n})$ is associated with a "clock" measuring the residual time until event s will occur. The lifetime of a clock associated with event s is a random variable drawn from a distribution with a continuous CDF $F_s(x)$ with $F_s(0) = 0$.

For $s \in \mathcal{S}_0$, $F_s(x)$ is exponential with mean λ_s^{-1} and for $s \in \mathcal{S}_1$, $F_s(x)$ is general with mean μ_s^{-1}. A special case is where $\mathcal{S}_0$ labels customer inter-arrival times and $\mathcal{S}_1$ labels service lifetimes. In general, $\mathcal{S}$ labels possible point events.

The GSMP is constructed as follows. The process $\mathbf{n}(t)$ evolves from any state $\mathbf{n}$ by having some event $s \in \mathcal{S}(\mathbf{n})$ trigger a transition to another state $\mathbf{n}'$. Let $p(\mathbf{n}'|s, \mathbf{n})$ be the probability that the new state is $\mathbf{n}'$ given that s triggers the transition. The actual triggering event depends on the clocks associated with the events of $\mathcal{S}(\mathbf{n})$ and the speeds at which they run, $c(\mathbf{n}, s)$.

When the state of the process is $\mathbf{n}$, $c(\mathbf{n}, s) = 0$ (inactive) for every $s \notin \mathcal{S}(\mathbf{n})$ and at least one $c(\mathbf{n}, s) > 0$ (active) for $s \in \mathcal{S}(\mathbf{n})$. A triggering event is also referred to as an expiration of its corresponding clock. It is assumed that no two active

clocks can expire simultaneously. However, an instantaneous reactivation of a clock is possible.

At a transition from $\mathbf{n}$ to $\mathbf{n}'$ triggered by event s, new clock values are independently drawn for $s' \in \mathcal{S}(\mathbf{n}') - \big(\mathcal{S}(\mathbf{n}) - \{s\}\big)$. The old active clocks' reading is kept after transition. It should be noted that the GSMP defined in [47] allows a clock lifetime distribution that depends on the tuple $(\mathbf{n}, s, \mathbf{n}', s')$ rather than only on s', i.e., a CDF $F_{\mathbf{n},s,\mathbf{n}',s'}(x)$.

It has been shown, e.g., in [48], that if the GSMP $\mathbf{n}(t)$ has a finite stationary distribution and the property of *instantaneous attention* (i.e., generally distributed lifetime events $\{s\}$ are activated with a positive rate as soon as they are created), then insensitivity is equivalent to stationary *partial balance* of the Markovian version of $\mathbf{n}(t)$ (i.e., when all $F_s(x)$ are exponentially distributed). That is, the following condition is necessary and sufficient for insensitivity.

Condition 1: With all event lifetimes set to exponential distributions, the stationary probability flux out of each state $\mathbf{n}$ due to a particular lifetime event clock expiration is equal to the stationary influx probability into that state due to an activation of that event clock.

It is worth noting that not every Markovian process satisfying partial balance can be embedded into a GSMP. One example which can be embedded is M/M/1 with processor sharing (PS). On the other hand, as far as we are aware of, the M/M/1 FIFO and LIFO queue cannot be embedded into a GSMP. Nevertheless, using the symmetric queue framework model of [37], it can be shown that M/G/1 LIFO queue is insensitive. On the other hand, M/G/1 FIFO is not a GSMP neither a symmetric queue and also known to be sensitive.

ACKNOWLEDGMENT

The authors wish to thank Peter G. Taylor for his insightful comments on the insensitivity of the I-J-FISF assignment policy. We also thank the anonymous reviewers for their valuable comments that contributed to the improved quality of this paper.

REFERENCES

[1] U.S. Environmental Protection Agency, "EPA report on server and data center energy efficiency," Tech. Rep., 2007.

[2] The Climate Group, "SMART 2020: Enabling the low carbon economy in the information age," Tech. Rep., 2008.

[3] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE FOCS '95*, Milwaukee, WI, Oct. 1995, pp. 374–382.

[4] N. Bansal, K. Pruhs, and C. Stein, "Speed scaling for weighted flow time," in *Proc. ACM-SIAM SODA '07*, New Orleans, LA, Jan. 2007, pp. 805–813.

[5] N. Bansal, H.-L. Chan, and K. Pruhs, "Speed scaling with an arbitrary power function," in *Proc. ACM-SIAM SODA '09*, New York, NY, Jan. 2009, pp. 693–701.

[6] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *Proc. IEEE INFOCOM '09*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2007–2015.

[7] L. L. H. Andrew, M. Lin, and A. Wierman, "Optimality, fairness and robustness in speed scaling designs," in *Proc. ACM SIGMETRICS '10*, New York, NY, Jun. 2010, pp. 37–48.

[8] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, to appear.

[9] F. Bonomi, "On job assignment for a parallel system of processor sharing queues," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 858–869, Jul. 1990.

[10] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributed web-server systems," *ACM Computing Surveys*, vol. 34, no. 2, pp. 263–311, Jun. 2002.

[11] V. Gupta, K. Sigman, M. Harchol Balter, and W. Whitt, "Insensitivity for PS server farms with JSQ routing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 2, pp. 24–26, Sep. 2007.

[12] V. Gupta, M. Harchol Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms," *Performance Evaluation*, vol. 64, pp. 1062–1081, Oct. 2007.

[13] E. Altman, U. Ayesta, and B. J. Prabhu, "Load balancing in processor sharing systems," *Telecommunication Systems*, vol. 47, pp. 35–48, Jun. 2011.

[14] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services," *Performance Evaluation*, vol. 68, no. 11, pp. 1056–1071, Nov. 2011.

[15] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[16] T. Bonald and A. Proutière, "Insensitivity in processor-sharing networks," *Performance Evaluation*, vol. 49, pp. 193–209, Sep. 2002.

[17] T. Bonald and A. Proutière, "Insensitive bandwidth sharing in data networks," *Queueing Systems*, vol. 44, no. 1, pp. 69–100, May 2003.

[18] V. B. Iversen, *Teletraffic Engineering and Network Planning*. Tech. Univ. Denmark, 2010.

[19] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman, "The case for RAMClouds: Scalable high-performance storage entirely in DRAM," *SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 92–105, Dec. 2009.

[20] A. M. Caulfield, L. M. Grupp, and S. Swanson, "Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications," *ACM SIGPLAN Notices*, vol. 44, no. 3, pp. 217–228, Mar. 2009.

[21] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2009.

[22] F. A. Haight, "Two queues in parallel," *Biometrika*, vol. 45, pp. 401–410, Dec. 1958.

[23] W. Winston, "Optimality of the shortest line discipline," *Journal of Applied Probability*, vol. 14, no. 1, pp. 181–189, Mar. 1977.

[24] R. R. Weber, "On the optimal assignment of customers to parallel servers," *Journal of Applied Probability*, vol. 15, no. 2, pp. 406–413, Jun. 1978.

[25] W. Whitt, "Deciding which queue to join: Some counterexamples," *Operations Research*, vol. 34, no. 1, pp. 55–62, Jan./Feb. 1986.

[26] Z. Liu and D. Towsley, "Optimality of the round-robin routing policy," *Journal of Applied Probability*, vol. 31, no. 2, pp. 466–475, Jun. 1994.

[27] E. Hyytiä, A. Penttinen, S. Aalto, and J. Virtamo, "Dispatching problem with fixed size jobs and processor sharing discipline," in *Proc. 23rd International Teletraffic Congress*, San Fransisco, CA, Sep. 2011, pp. 190–197.

[28] E. Hyytiä, J. Virtamo, S. Aalto, and A. Penttinen, "M/M/1-PS queue and size-aware task assignment," *Performance Evaluation*, vol. 68, no. 11, pp. 1136–1148, Nov. 2011.

[29] E. Koenigsberg, "On jockeying in queues," *Management Science*, vol. 12, no. 5, pp. 412–436, Jan. 1966.

[30] Y. Zhao and W. K. Grassmann, "The shortest queue model with jockeying," *Naval Research Logistics*, vol. 37, no. 5, pp. 773–787, Oct. 1990.

[31] I. Adan, J. Wessels, and W. Zijm, "Analysis of the asymmetric shortest queue problem with threshold jockeying," *Stochastic Models*, vol. 7, no. 4, pp. 615–627, 1991.

[32] I. Adan, J. Wessels, and W. Zijm, "Matrix-geometric analysis of the shortest queue problem with threshold jockeying," *Operations Research Letters*, vol. 13, no. 2, pp. 107–112, Mar. 1993.

[33] Y. Zhao and W. K. Grassmann, "Queueing analysis of a jockeying model," *Operations Research*, vol. 43, no. 3, pp. 520–529, May/Jun. 1995.

[34] Y. Sakuma, "Asymptotic behavior for MAP/PH/c queue with shortest queue discipline and jockeying," *Operations Research Letters*, vol. 38, no. 1, pp. 7–10, Jan. 2010.

[35] M. Bramson, Y. Lu, and B. Prabhakar, "Randomized load balancing with general service time distributions," in *Proc. ACM SIGMETRICS '10*, New York, NY, Jun. 2010, pp. 275–286.

[36] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization," *ACM Transactions on Algorithms*, vol. 3, no. 4, pp. 49:1–49:17, Nov. 2007.

[37] F. P. Kelly, "Networks of queues," *Advances in Applied Probability*, vol. 8, no. 2, pp. 416–432, Jun. 1976.

[38] A. D. Barbour, "Networks of queues and the method of stages," *Advances in Applied Probability*, vol. 8, no. 3, pp. 584–591, Sep. 1976.

[39] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. 2.

[40] S. Stidham, Jr. and R. R. Weber, "Monotonic and insensitive optimal policies for control of queues with undiscounted costs," *Operations Research*, vol. 37, no. 4, pp. 611–625, Jul./Aug. 1989.

[41] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Operations Research*, vol. 49, no. 5, pp. 720–731, Sep./Oct. 2001.

[42] S. A. Lippman, "Applying a new device in the optimization of exponential queuing systems," *Operations Research*, vol. 23, no. 4, pp. 687–710, Jul./Aug. 1975.

[43] J. Wijngaard and S. Stidham, Jr., "Forward recursion for Markov decision processes with skip-free-to-the-right transitions, Part I: Theory and algorithm," *Mathematics of Operations Research*, vol. 11, no. 2, pp. 295–308, May 1986.

[44] K. Matthes, "Zur Theorie der Bedienungsprozesse," in *Proc. 3rd Prague Conf. Information Theory, Statistical Decision Functions and Random Processes*, Prague, 1962.

[45] R. Schassberger, "Insensitivity of steady-state distributions of generalized semi-Markov processes. Part I," *The Annals of Probability*, vol. 5, no. 1, pp. 87–99, Feb. 1977.

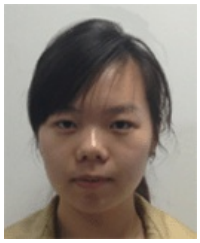
[46] R. Schassberger, "Insensitivity of steady-state distributions of generalized semi-Markov processes. Part II," *The Annals of Probability*, vol. 6, no. 1, pp. 85–93, Feb. 1978.

[47] W. Whitt, "Continuity of generalized semi-Markov processes," *Mathematics of Operations Research*, vol. 5, no. 4, pp. 494–501, Nov. 1980.

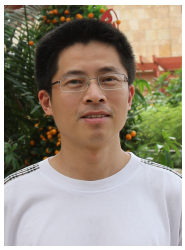
[48] R. Schassberger, "Two remarks on insensitive stochastic models," *Advances in Applied Probability*, vol. 18, no. 3, pp. 791–814, Sep. 1986.

Zvi Rosberg received the B.Sc., M.A. and Ph.D. degrees from the Hebrew University of Jerusalem. Currently, he is a visiting professor at the EE department of City University of Hong Kong. He serves on the editorial board of the Wireless Networks (WINET). His research interest include networking, traffic management, wireless communication, optical networks, Internet technologies, transmission rate and power control in cellular networks, scheduling and routing, network flow control, control in queueing networks, analysis of algorithms and performance evaluation. Previously held positions in Academia include: CORE, Catholic University of Louvain, Belgium; University of Illinois, USA; Technion-IIT, Israel; KTH, Sweden; Ben-Gurion University, Israel; and in the industry: Radware Ltd., IBM Research Lab. and CSIRO ICT, Australia.

Yu Peng received the B.Eng. degree in information engineering from City University of Hong Kong, Hong Kong, in 2010. He is currently working toward the Ph.D. degree in the department of Electronic Engineering at City University of Hong Kong. His research interests include teletraffic theory, optical network dimensioning and performance evaluation of telecommunication networks.



Jing Fu received the B.Eng. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2011. She is currently working toward the Ph.D. degree in the Department of Electronic Engineering, City University of Hong Kong. Her research interest is now focused on energy efficiency in server farms.



Jun Guo (S'01–M'06) received the B.E. degree in automatic control engineering from Shanghai University of Science and Technology, Shanghai, China, in 1992 and the M.E. degree in telecommunications engineering and the Ph.D. degree in electrical and electronic engineering from the University of Melbourne, Melbourne, Australia, in 2001 and 2006, respectively. He was with the School of Computer Science and Engineering, The University of New South Wales, Australia as a senior research associate from 2006 to 2008 and on an Australian Postdoctoral Fellowship (APD) supported by the Australian Research Council (ARC) from 2009 to 2011. Since March 2012, he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is now a research fellow. His research is currently focused on teletraffic theory and its applications in service sectors, multicast in wired/wireless networks, and multihoming in heterogeneous access networks.

Eric W. M. Wong (S'87–M'90–SM'00) received the B.Sc. and M.Phil. degrees in electronic engineering from the Chinese University of Hong Kong, Hong Kong, in 1988 and 1990, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1994. In 1994, he joined the City University of Hong Kong, where he is now an Associate Professor with the Department of Electronic Engineering. His research interests include the analysis and design of telecommunications networks, optical switching and video-on-demand systems.

Moshe Zukerman (M'87–SM'91–F'07) received the B.Sc. degree in industrial engineering and management and the M.Sc. degree in operations research from the Technion – Israel Institute of Technology, Haifa, Israel, and the Ph.D. degree in engineering from University of California, Los Angeles, in 1985. He was an independent Consultant with the IRI Corporation and a Postdoctoral Fellow with the University of California, Los Angeles, in 1985–1986. In 1986–1997, he was with the Telstra Research Laboratories (TRL), first as a Research Engineer and, in 1988–1997, as a Project Leader. He also taught and supervised graduate students at Monash University in 1990–2001. During 1997–2008, he was with The University of Melbourne, Victoria, Australia. In 2008 he joined City University of Hong Kong as a Chair Professor of Information Engineering, and a group leader. He has served on various editorial boards such as Computer Networks, IEEE Communications Magazine, IEEE Journal of Selected Ares in Communications, IEEE/ACM Transactions on Networking and the International Journal of Communication Systems.