On the impact of heterogeneity and back-end scheduling in load balancing designs

Ho-Lin Chen
Information Science and Technology
California Institute of Technology
Pasadena, CA 91125

Jason R. Marden
Information Science and Technology
California Institute of Technology
Pasadena, CA 91125

Adam Wierman
Computer Science Department
California Institute of Technology
Pasadena, CA 91125

Abstract—**Load balancing is a common approach for task assignment in distributed architectures. In this paper, we show that the degree of inefficiency in load balancing designs is highly dependent on the scheduling discipline used at each of the back-end servers. Traditionally, the back-end scheduler can be modeled as Processor Sharing (PS), in which case the degree of inefficiency grows linearly with the number of servers. However, if the back-end scheduler is changed to Shortest Remaining Processing Time (SRPT), the degree of inefficiency can be independent of the number of servers, instead depending only on the heterogeneity of the speeds of the servers. Further, switching the back-end scheduler to SRPT can provide significant improvements in the overall mean response time of the system as long as the heterogeneity of the server speeds is small.**

I. Introduction

Load balancing is a common approach to task assignment in computer communication systems such as web server farms, database systems, and grid computing clusters. In such designs there is a dispatcher that seeks to balance the assignment of service requests (jobs) across the back-end servers in the system so that the response time of jobs at each server is (nearly) the same. Such designs are popular due to the increased robustness they provide to bursts of traffic, server failures, etc., as well as the inherent scalability they provide. However, there is also a major drawback to load balancing designs – some performance is sacrificed. Specifically, it is possible to reduce user response times by moving away from load balancing designs.

In this paper, we study the impact of the scheduler at the back-end servers, i.e., the back-end scheduler. Our goal is two-fold: (i) we will characterize the inefficiency of load balancing designs and determine how this inefficiency depends on the back-end scheduler; and (ii) we will study the improvements in overall mean response time that are achievable through changing the back-end scheduler.

Our results apply to a wide variety of back-end schedulers, however our primary focus will be on two particularly important options: Processor Sharing (PS) and Shortest Remaining Processing Time first (SRPT). Under PS the server is shared evenly among all jobs at the server, while under SRPT the server is devoted fully to the job with the least remaining work. PS is often used as a simplified model of the traditional scheduling designs in many computer systems including bandwidth scheduling in web servers, flow scheduling in routers, and CPU scheduling in operating systems. SRPT provides an important comparison because it has recently been suggested as an alternative to PS in a variety of applications, e.g., [11], [14], [16], [18], [23]. Further, SRPT optimizes mean response time in a single server queue [26]. However, it should be pointed out that SRPT is not necessarily optimal in load balancing systems since the arrival process to the back-end server is dependent on the queue state.

There are three main contributions in this paper.

First, we show that the back-end scheduler has a significant impact on the degree of inefficiency in load balancing designs. In particular, when the back-end scheduler is PS the degree of inefficiency depends linearly on the parallelism of the system (i.e., the number of servers) but when the back-end scheduler is SRPT the degree of inefficiency is less dependent on the system parallelism and more dependent on the heterogeneity of the speed of the servers (Theorems 4 and 5). In fact, in the case when job sizes follow a Pareto distribution with infinite variance, the degree of inefficiency under load balancing systems is independent of the parallelism of the system and linearly dependent on the heterogeneity of the system.

Second, we illustrate that the potential improvement from changing the back-end scheduler can be dramatic; however, the degree of improvement depends on the heterogeneity of the server speeds (Corollary 3). In particular, the improvement in mean response time from switching from PS to SRPT can be as dramatic as the improvement in a simple single server queue when the heterogeneity of server speeds is small. However, when the heterogeneity of server speeds is large, the improvement can be very small or even non-existent.

Third, we provide results that facilitate the analysis of arbitrary back-end schedulers in load balancing designs (Theorems 6 and 7). These results provide tools that are straightforward to apply to determine the performance of an arbitrary scheduling policy in a load balancing system under a general service distribution.

Our results are enabled via two analytic techniques: (i) algorithmic game theory and (ii) heavy-traffic approximations. In particular, the analysis begins by noting that a load balancing dispatcher can be viewed as equivalent to having jobs make their own dispatching decisions greedily in order to minimize their mean response time. Thus an ideal load balancer can

This work was supported by NSF CCF 0830511, Microsoft Research, and the Lee Center for Advanced Networking.

be viewed as running at an operating point determined by the Nash equilibrium of a selfish routing game [19]. So, we can bring to bear tools from the algorithmic game theory community, and further, the inefficiency of load balancing designs can be measured by the so-called "price of anarchy" (see Section II for the definition). However, even in this framework, understanding the exact interaction between the back-end scheduler and the load balancer is difficult (as we highlight in Section IV). Thus, we consider the heavy-traffic regime, where the load of the system is approaching 1, and we take advantage of recent results, e.g., [5], [30], [32], to simplify the model considerably. Further, the heavy-traffic regime is the regime of interest since web applications are typically run at high load and, additionally, the inefficiency of load-balancing designs is intuitively worst under heavy-traffic.

The remainder of the paper is organized as follows. We first formally introduce our model in Section II. Then in Section III we summarize the prior literature. Next in Section IV we provide the necessary background on PS and SRPT. Section V presents the first set of new results, which contrast the performance of PS and SRPT back-end schedulers under Pareto job sizes. Then, Section VI presents results characterizing the performance of arbitrary back-end schedulers under general job size distributions. Finally, Section VII concludes the paper.

II. Model description

To study the efficiency of load balancing designs, we will use the queueing model pictured in Figure 1. The system consists of n parallel queues $Q_1, \ldots, Q_n$ with service rates $\mu_1, \ldots, \mu_n$ where $\mu_i \geq \mu_{i+1}$ and $\mu_i = k_i\mu$. Let X_i be a random i.i.d. job size at queue i having p.d.f. $f_i(x)$ and c.d.f. $F_i(x)$. Let $\overline{F}_i(x) = 1 - F_i(x)$ and $E[X_i] := 1/\mu_i$.

The arrival process to the system is Poisson with rate Λ, where $\Lambda < \sum_{i=1}^n \mu_i$ ensures stability. There is a load balancing dispatcher that probabilistically routes arrivals to queues so that the mean response time (a.k.a. sojourn time, flow time), $E[T_i]$, at each queue i is the same. This model follows from the assumption that routing decisions are made without observing the queue length and that the load balancing dispatcher reacts to periodic performance measurements attained from each queue with the goal of balancing response times across the queues. This knowledge limitation of the dispatcher is a common design in distributed web servers.

It follows that the resulting arrival rate to Q_i is Poisson with rate λ_i and the load at queue i is $\rho_i := \lambda_i/\mu_i < 1$. Thus, each Q_i is a stationary M/GI/1 queue. Note $\Lambda = \sum_{i=1}^n \lambda_i$. Further, define the remaining service capacity, a.k.a., "gap", at Q_i as

$$\gamma_i := \mu_i - \lambda_i$$
$$\Gamma := \sum_{i=1}^{n} \gamma_i.$$

This will be an important concept in our analysis. Finally, note that all incoming jobs are routed to one of the servers, i.e., there is no balking.

For reasons that we will describe in Section IV, *we will be considering the heavy-traffic behavior of this system.* That is,

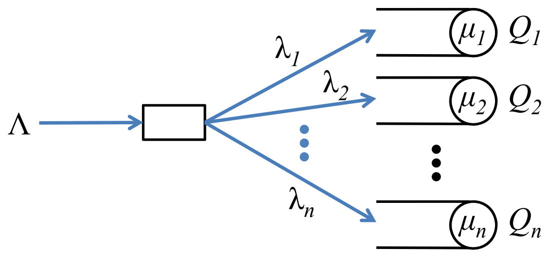

Fig. 1. A diagram of the load balancing model considered in this paper.

we will analyze the response time as $\Lambda \to \sum_{i=1}^n \mu_i$, which also ensures that each Q_i is in heavy-traffic, i.e. $\rho_i \to 1$. We do not view this as a limitation since the worst-case inefficiency of load balancing designs intuitively occurs in heavy-traffic. Further, the heavy-traffic regime is the most relevant scenario for web applications.

Interestingly, we can take a slightly different view of the load balancing model than we have described to this point which will prove useful. In particular, we can view the stationary behavior of the load balancing system as the equilibrium point of a non-atomic routing game. This is often termed a selfish routing game [19]. A set $\{\lambda_1, \ldots, \lambda_n\}$ corresponds to Nash equilibrium arrival rates (a.k.a., the Nash assignment) in this non-atomic routing game if for all Q_i, if $\lambda_j > 0$, then $E[T_i] \leq E[T_j]$. Informally, a selfish infinitesimal of arriving flow cannot improve its mean response time if the system is at a Nash equilibrium. Note that since a load balancing dispatcher maintains the same $E[T_i]$ for all Q_i such that $\lambda_i > 0$, it is operating at a Nash equilibrium. Further, this Nash equilibrium is unique.

In the setting of a selfish routing game, the *price of anarchy* (PoA) is defined as the worst-case ratio between a Nash assignment and the global optimum. Formally, we write

$$\begin{aligned} &\max \frac{E[T; (\lambda_1^{ne}, \ldots, \lambda_n^{ne}), k]}{E[T; (\lambda_1^{opt}, \ldots, \lambda_n^{opt}), k]} \\ &\text{s.t. } 0 \leq \{\lambda_i^{ne}, \lambda_i^{opt}\} < \mu_i, \quad 1 \leq i \leq n \\ &\qquad \mu \leq \mu_i \leq k\mu, \quad 1 \leq i \leq n, \end{aligned}$$

where λ_i^{opt} corresponds to the optimal routing structure and λ_i^{ne} corresponds to the Nash assignment (load balancing routing structure). Note the Nash and optimal assignments are both unique in non-atomic routing games. We have inserted a parameter k that bounds the ratio of the server speeds. We will refer to k as the "heterogeneity" of the system and we will state bounds on the price of anarchy in terms of the number of servers, n, and the heterogeneity, k.

In our context, the price of anarchy is equivalent to the efficiency of a load balancing design. Interestingly, in addition the standard interpretation, the price of anarchy takes on another meaning in this context – it also characterizes the benefit achievable by switching from PS to SRPT (see the discussion of Corollary 3 for details).

III. Prior work

Non-atomic routing games were first introduced by Pigou [22] and later were formally defined by Wardrop [28]. For this reason, equilibria in these games are often called "Wardrop equilibria." The price of anarchy was first studied in this setting by Roughgarden & Tardos [25], and the work that followed is surveyed in [19]. The fundamental result for non-atomic routing games is that the price of anarchy is independent of the network structure under a wide variety of latency (response time) functions, e.g., under linear latency functions the price of anarchy is bounded by 4/3 regardless of the network structure considered [25].

However, there are very few results in the case of latency functions specified by queueing models, as we consider in this paper. In particular, the only bound on the price of anarchy under queueing latency functions that is known to hold independently of the network structure holds only when $\Lambda < \min_i \mu_i$, see [24]. In this case the price of anarchy is bounded by

$$\frac{1}{2}\left(1+\sqrt{\frac{\min_i(\mu_i)}{\min_i(\mu_i)-\Lambda}}\right).$$

The reason such results only hold in this limited situation is that outside of the light-traffic regime the network structure matters. Recently, the first result to characterize the impact of the network structure outside of the light-traffic regime was provided independently by Haviv & Roughgarden [13] and Wu & Starobinski [36] who proved that, in the model of this paper the price of anarchy is n, the number of queues in the system, when job sizes are Exponentially distributed and jobs are processed in FCFS order. In the current paper our goal is to contrast the price of anarchy under a variety of other back-end scheduling disciplines, particularly the price of anarchy under PS and SRPT.

Our analysis depends primarily on characterizing the heavy-traffic behavior of scheduling disciplines. There is a large literature studying queueing systems in heavy-traffic, e.g. [7], [12], [33], [38]. Some of these results have been exploited to study distributed system designs that use dispatchers such as Round Robin (RR), Join the Shortest Queue (JSQ), and SITA-E. See [9], [34], [35] and the references therein. However, our focus in this paper differs from these analyses for a number of reasons. First, we focus on characterizing the inefficiency of load balancing dispatchers rather than simply deriving the performance of load balancing dispatchers (though we derive the performance as a side-effect). Second, we derive results that apply when the back-end scheduler is SRPT, which has not been studied previously in this context. Third, our results apply for arbitrary back-end schedulers. Specifically, if the heavy-traffic behavior of the policy is known our technique can be applied to achieve results. The works most closely related to ours are the recent papers by Wu & Down [9], [34] which study the heavy-traffic behavior of approximations of SRPT in distributed architectures where the dispatcher performs a multi-layered round robin algorithm.

IV. Background on back-end schedulers

We will primarily consider two possibilities for the back-end scheduler, PS and SRPT, though we will also discuss other scheduling disciplines in Section VI. PS is important

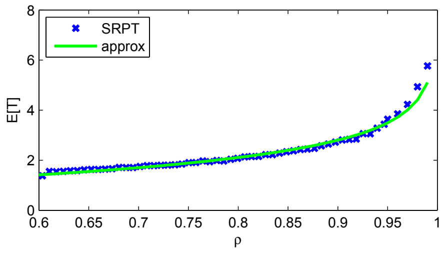

Fig. 2. An illustration of the validity of the approximation in (1).

because it is commonly used as a tractable model for the traditional scheduling designs in many computer systems including, bandwidth scheduling in web servers, flow scheduling in routers, and CPU scheduling in operating systems. SRPT is important because it is known to minimize the mean response time in a single server queue [26] regardless of the arrival and service process. Further, it has recently been suggested as an alternative to PS in many applications [11], [17], [23]. Note however that SRPT is not necessarily the optimal back-end scheduler for a load balancing design.[1] In particular, because the load balancer interacts with the scheduling policy, it is not a priori clear that SRPT is even necessarily a good choice for a back-end scheduler.

There is a large literature studying each of these policies and we refer the reader to [37] for background on PS and to [15] for background on SRPT. For our purposes, we will need only results characterizing the mean response time under these policies. The interested reader may find surveys of distributional results for SRPT and PS in [6], [21] and studies of the fairness of SRPT in [3], [29].

A. Processor Sharing (PS)

In an M/GI/1 PS queue $E[T_i]$ is [15]: $E[T_i] = \frac{1}{\mu_i - \lambda_i}$. The important observation here is that $E[T]^{M/GI/1/PS} = E[T]^{M/M/1/FCFS}$. Thus, the price of anarchy under PS follows immediately from prior results [13], [36]:

Proposition 1. *The price of anarchy in a distributed system with Poisson arrivals when the back-end scheduler is PS is n.*

Further, [13] illustrates that this is tight. Recently, this result has been extended to multi-class load balancing systems [1].

B. Shortest Remaining Processing Time (SRPT)

The mean response time under SRPT has a much more complicated form. Let $m_{ij}(x) = E[X_i^j 1_{X_i<x}] = \int_0^x t^j f_i(t)dt$ be a truncated j-th moment of job size distribution X_i. Let $\widetilde{m_{ij}}(x) = E[\min(x, X_i^j)] = i\int_0^x t^{j-1}\overline{F}_i(t)dt$ be a different truncated j-th moment. Finally, let $\rho_i(x) = \lambda_i m_{i1}(x)$. Now, we can write the mean response time of SRPT in an M/GI/1 queue as follows [27]:

$$E[T_i] = \int_0^\infty \left(\int_0^x \frac{1}{1-\rho_i(t)}dt + \frac{\lambda_i \widetilde{m_{i2}}(x)}{2(1-\rho_i(x))^2}\right) dF_i(x).$$

[1]If the dispatcher choices are fixed, then SRPT is the optimal back-end scheduler. However, since the dispatcher's decisions depend on the back-end scheduler, SRPT is not optimal.

The complicated form of $E[T_i]$ under SRPT makes it difficult to study this policy directly. Instead, we will consider the behavior of this policy in *heavy-traffic*, i.e., as $\Lambda \to \sum_{i=1}^{n} \mu_i$. In this case, recent results provide a simpler form under certain job size distributions including bounded distributions [30], [32], exponential distributions [4], and Pareto distributions [5], [31]. For the bulk of the paper we will limit ourselves to Pareto job size distributions, since these distributions are commonly found to be good models of web request distributions, e.g., see [2], [8], [10]. However, in Section VI, we will discuss other job size distributions.

The following proposition follows from combining the results in [5], [31].

Proposition 2. *Consider an M/GI/1 SRPT queue, as* $\lambda \to \mu$, *and* $X \sim Pareto(\alpha, x_L)$ *with* $\alpha > 1$, *i.e.,* $\overline{F}(x) = (x/x_L)^{\alpha}$ *for some* $x_L > 0$, *then*

$$E[T] = \begin{cases} \Theta\left(\log\left(\frac{1}{1-\rho}\right)\right), & \text{if } \alpha < 2 \\ \Theta\left(\log^2\left(\frac{1}{1-\rho}\right)\right), & \text{if } \alpha = 2 \\ \Theta\left(\frac{1}{(1-\rho)^{\frac{\alpha-2}{\alpha-1}}}\right), & \text{if } \alpha > 2. \end{cases}$$

Due to limited space, we will focus only on the case of $\alpha \neq 2$, though the case of $\alpha = 2$ can be handled using the same techniques. Note that when $\alpha < 2$ the job size variance is infinite.

Proposition 2 only specifies the "growth rate" of $E[T_i]$ with ρ_i under SRPT in heavy-traffic, and we need a simple equation to facilitate our analysis. So, we will use the following functional form that encompasses both $E[T_i]$ under PS and an approximation for $E[T_i]$ under SRPT in heavy-traffic.

$$E[T_i] := \begin{cases} \frac{1}{\mu_i} \log\left(\frac{\mu_i}{\mu_i - \lambda_i}\right), & \text{if } \alpha < 2 \\ \frac{1}{\mu_i}\left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m, & \text{if } \alpha > 2, \end{cases} \tag{1}$$

where $0 < m = \frac{\alpha-2}{\alpha-1} < 1$ ($m = 1$ under PS). This approximation can been shown using simulations to be very accurate for SRPT, even outside of heavy-traffic. Figure 2 illustrates this fact by comparing the approximation to results from a simulation of an M/GI/1 queue with a $Pareto(1.2)$ job size distribution. Further, the approximation matches the bounds on SRPT derived in [31].

Under this formulation, we see that the contribution of Q_i to the overall response time is given by $\lambda_i E[T_i]$ which, in heavy-traffic ($\lambda_i/\mu_i \to 1$), becomes

$$C_i(\lambda_i) := \begin{cases} \log\left(\frac{\mu_i}{\mu_i - \lambda_i}\right), & \text{if } \alpha < 2 \\ \left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m, & \text{if } \alpha > 2 \end{cases}$$

For a given set of arrival rates, $(\lambda_1, ..., \lambda_n)$, the expected overall response time of the system in heavy traffic is

$$E[T; (\lambda_i, \dots, \lambda_n)] := \sum_{i=1}^{n} C_i(\lambda_i).$$

V. The case of SRPT and Pareto job sizes

We begin our analysis by focusing on the inefficiency of load balancing designs when job sizes follow a Pareto job size distribution and the back-end scheduler performs SRPT. Table IV-B summarizes the main results from this section.

Let us first concentrate on the price of anarchy results for SRPT. Notice that when the job sizes are Pareto with $\alpha < 2$ (infinite variance) the price of anarchy is k. That is, it depends linearly on the heterogeneity of server speeds (k) and is independent of the parallelism of the system (n). This is in stark contrast with the result for PS, which states that the price of anarchy grows linearly with the number of servers (i.e., is n). Thus, it becomes evident that the overall design of a load balancing system should be dependent on the back-end scheduler being used. That is, if one is using PS it is important to avoid adding too many servers, while under SRPT it is primarily important to limit the variation between the servers.

When we switch to considering SRPT under Pareto jobs sizes with $\alpha > 2$, we see the same contrast with PS, only to a lesser extent. Again, the parallelism of the system is less important for the inefficiency of SRPT load balancing systems than for PS ones, but the difference shrinks as job size variability decreases (α, m increase). It should be pointed out that the price of anarchies of SRPT and PS are tight. In particular, they are tight when $\mu_1 = k\mu$ and $\mu_2 = \ldots = \mu_n = \mu$.

It is important to remember that these results characterize the price of anarchy of SRPT under heavy-traffic. However, intuitively, the heavy-traffic regime should provide the worst-case price of anarchy. Our results verify that this is indeed true under PS since the price of anarchy in heavy-traffic matches the overall price of anarchy.

Interestingly, the price of anarchy has a second interpretation in the context of this paper. This interpretation is the result of the following corollary, which follows from the results for the Nash and optimal assignments stated in Table IV-B.

Corollary 3. λ_i^{opt} *under SRPT with Pareto job sizes having* $\alpha < 2$ *is the same as* λ_i^{ne} *under PS.*

The importance of this Corollary is not immediately evident, however it provides an alternative interpretation of the price of anarchy in the case of Pareto job sizes with $\alpha < 2$. In particular, in this case the price of anarchy bounds the reduction of the benefit attainable by switching the back-end scheduler from PS to SRPT (as compared to the benefit in an M/GI/1 queue). To see this, note that because λ_i^{opt} under SRPT is the same as λ_i^{ne} under PS, the ratio between $E[T^{opt}]$ under SRPT and $E[T^{ne}]$ under PS is the same as in the M/GI/1 queue. Further, the price of anarchy of SRPT characterizes how much worse $E[T^{ne}]$ is than $E[T^{opt}]$ under SRPT. So, the price of anarchy bounds the reduction in the improvement from switching to SRPT as compared to the improvement in the M/GI/1.

Applying this interpretation of the price of anarchy, it is clear that a dramatic improvement is possible by switching from PS to SRPT in load balancing systems when k is small, i.e., since the improvement of SRPT over PS in the M/GI/1 is large, e.g., see [31], it is still large in load balancing systems

TABLE I
SUMMARY OF RESULTS FOR PARETO JOB SIZES. THE RESULTS FOR PS WERE FIRST DERIVED IN [13], [36], BUT ALSO FOLLOW IMMEDIATELY FROM THE RESULTS FOR SRPT.

Back-end scheduler	**SRPT**		**PS**
Job size distribution	**Pareto with $\alpha < 2$**	**Pareto with $\alpha > 2$**	**General**
$C_i(\lambda_i)$	$\log\left(\frac{\mu_i}{\mu_i-\lambda_i}\right)$	$\left(\frac{\mu_i}{\mu_i-\lambda_i}\right)^m$ where $m = \frac{\alpha-2}{\alpha-1}$	$\frac{1}{\mu_i-\lambda_i}$
Optimal assignment, λ_j^{opt}	$\mu_j - \left(\frac{1}{n}\right)\Gamma$ Lemma 8	$\mu_j - \left(\frac{\mu_j^{m/(m+1)}}{\sum_{i=1}^n \mu_i^{m/(m+1)}}\right)\Gamma$ Lemma 9	$\mu_j - \left(\frac{\sqrt{\mu_j}}{\sum_{i=1}^n \sqrt{\mu_i}}\right)\Gamma$ Lemma 9
Nash assignment, λ_j^{ne}	Satisfy $1 - \frac{\lambda_j^{ne}}{\mu_j} = \left(1 - \frac{\lambda_1^{ne}}{\mu_1}\right)^{\mu_j/\mu_1}$ with λ_1^{ne} s.t. $\Gamma = \sum_{i=1}^n \mu_i \left(1 - \frac{\lambda_1^{ne}}{\mu_1}\right)^{\mu_i/\mu_1}$ Lemma 10	$\mu_j - \left(\frac{\mu_j^{(m-1)/m}}{\sum_{i=1}^n \mu_i^{(m-1)/m}}\right)\Gamma$ Lemma 11	$\mu_j - \left(\frac{1}{n}\right)\Gamma$ Lemma 11
Price of anarchy	k **Theorem 4**	$k^{1-m}n^m$ **Theorem 5**	n **Proposition 1**

when k is small. However, when k is large, PS can actually outperform SRPT.

In the remainder of this section, we prove the price of anarchy results in Table IV-B. To prove these results, we first need to characterize the λ_i^{ne} and λ_i^{opt}. These results are summarized in Table IV-B, but we defer the derivations to the appendix. We start with the case of infinite-variance Pareto job sizes and then move to the case of finite-variance.

Theorem 4. *When $C_i(\lambda_i) = \log\left(\frac{\mu_i}{\mu_i-\lambda_i}\right)$ the price of anarchy is k.*

Proof: We begin by proving an upper bound on the price of anarchy, and then we illustrate that it is asymptotically tight. Since the form of λ_i^{ne} is implicit, we cannot simply directly compare $E[T^{ne}]$ and $E[T^{opt}]$. Instead, we will write each in terms of the remaining service capacity at each queue, i.e., the gaps $\gamma_j = \mu_j - \lambda_j$.

Define c to be the average response time for the j-th queue $E[T_j^{ne}]$ (notice that every queue has the same average response time under Nash equilibrium). Then, we have

$$E[T^{ne}] = c\sum_{i=1}^n \mu_i.$$

Note that the remaining capacity at each server Q_j at the Nash assignment is

$$\gamma_j^{ne} = \frac{\mu_j}{e^{c\mu_j}}.$$

Next, we will calculate $E[T^{opt}]$ in terms of c and γ_j^{ne}. Since the total gap is distributed equally under the optimal allocation (see Table IV-B), we have

$$\gamma_j^{opt} = \frac{\Gamma}{n} = \frac{1}{n}\sum_{i=1}^n \frac{\mu_i}{e^{c\mu_i}}.$$

Thus, recalling that $\mu_i = k_i\mu$, we have that $E[T^{opt}]$ is as follows

$$\begin{aligned}
E[T^{opt}] &= \sum_{j=1}^n \log\left(\frac{\mu_j n}{\sum_{i=1}^n \frac{\mu_i}{e^{c\mu_i}}}\right) \\
&= \sum_{j=1}^n \log\left(\frac{k_j n}{\sum_{i=1}^n \frac{k_i}{e^{ck_i\mu}}}\right) \\
&= n\log(n) + \sum_{i=1}^n \log(k_i) + n\log\left(\frac{1}{\sum_{i=1}^n \frac{k_i}{e^{ck_i\mu}}}\right).
\end{aligned}$$

Noting that $k_i e^{-ck_i\mu}$ is decreasing for large enough $c\mu$, we can bound the third term above as follows:

$$\begin{aligned}
n\log\left(\frac{1}{\sum_{i=1}^n \frac{k_i}{e^{ck_i\mu}}}\right) &\geq n\log\left(e^{c\mu}/n\right) \\
&= nc\mu - n\log(n),
\end{aligned}$$

which gives

$$E[T^{opt}] \quad \geq \sum_{i=1}^n \log(k_i) + nc\mu.$$

Now, we can bound the price of anarchy by

$$\begin{aligned}
\frac{c\sum_{i=1}^n \mu_i}{\sum_{i=1}^n \log(k_i) + nc\mu} &= \frac{\sum_{i=1}^n k_i}{\frac{\sum_{i=1}^n \log(k_i)}{c\mu} + n} \\
&\leq \frac{\sum_{i=1}^n k_i}{n} \\
&\leq k,
\end{aligned}$$

Next, we will show that this bound on the price of anarchy is asymptotically tight. Consider the specific example where $\mu_1 = k\mu$ and $\mu_2 = \ldots = \mu_n = \mu$. Then, we can again calculate $E[T^{ne}]$ as

$$E[T^{ne}] = c\sum_{i=1}^n \mu_i = c\mu(k + n - 1),$$

and $E[T^{opt}]$ as

$$\begin{aligned} E[T^{opt}] &= \sum_{j=1}^{n} \log\left(\frac{\mu_j n}{\sum_{i=1}^{n} \frac{\mu_i}{e^{c\mu_i}}}\right) \\ &= (n-1)\log\left(\frac{\mu n}{\sum_{i=1}^{n} \frac{\mu_i}{e^{c\mu_i}}}\right) + \log\left(\frac{k\mu n}{\sum_{i=1}^{n} \frac{\mu_i}{e^{c\mu_i}}}\right) \\ &= n\log(\mu n) + \log(k) + n\log\left(\frac{1}{\sum_{i=1}^{n} \frac{\mu_i}{e^{c\mu_i}}}\right). \end{aligned}$$

Since $\mu_i \leq e^{c\mu_i}$ for large enough c (c can be chosen arbitrarily large in heavy-traffic), we can bound the last term above by

$$n\log\left(\frac{1}{\sum_{i=1}^{n} \frac{\mu_i}{e^{c\mu_i}}}\right) \leq n\log\left(e^{c\mu}\right) = nc\mu.$$

When considering the heavy-traffic regime, i.e., $c \to \infty$, this yields the following lower bound on the price of anarchy:

$$\begin{aligned} \frac{c\mu(k+n-1)}{n\log(\mu n) + \log(k) + nc\mu} &= \frac{k+n-1}{\frac{n\log(\mu n)+\log(k)}{c\mu} + n} \\ &\sim k/n \quad \text{as } c \to \infty. \end{aligned}$$

For constant n, this gives $\Omega(k)$ as desired. ■

Theorem 5. *When $C_i(\lambda_i) = \left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m$ the price of anarchy is $k^{1-m}n^m$.*

Proof: Since λ_i^{ne} and λ_i^{opt} are so similar (see Table IV-B), we can compare the mean response time under these policies directly. In particular, straightforward calculation yields

$$E[T;(\lambda_1^{opt},\ldots,\lambda_n^{opt}),k] = \left(\frac{1}{\Gamma}\right)^m \left(\sum_{j=1}^{n} \mu_j^{m/(m+1)}\right)^{m+1}$$

$$E[T;(\lambda_1^{ne},\ldots,\lambda_n^{ne}),k] = \left(\frac{1}{\Gamma}\right)^m \left(\sum_{j=1}^{n} \mu_j\right)\left(\sum_{j=1}^{n} \mu_j^{\frac{m-1}{m}}\right)^m,$$

which gives that the price of anarchy is the solution to the following optimization:

$$\begin{aligned} &\max \frac{\left(\sum_{j=1}^{n} k_j\right)\left(\sum_{j=1}^{n} k_j^{(m-1)/m}\right)^m}{\left(\sum_{j=1}^{n} k_j^{m/(m+1)}\right)^{m+1}} \\ &\text{s.t. } 1 \leq k_j \leq k. \end{aligned}$$

An upper bound on the solution to this optimization is the following reformulation:

$$\begin{aligned} &\max \frac{\left(\sum_{j=1}^{n} k_j\right)\left(\sum_{j=1}^{n} x_j^{1/m}\right)^m}{\left(\sum_{j=1}^{n} (k_j x_j)^{1/(m+1)}\right)^{m+1}} \\ &\text{s.t. } 1 \leq k_j \leq k \\ &\quad k^{m-1} \leq x_j \leq 1. \end{aligned}$$

We can write the above more succinctly using norms as follows:

$$\begin{aligned} &\max \frac{\|k_j\|_1 \|x_j\|_{1/m}}{\|k_j x_j\|_{1/(m+1)}} \\ &\text{s.t. } 1 \leq k_j \leq k \\ &\quad k^{m-1} \leq x_j \leq 1. \end{aligned}$$

where $\|y_j\|_p = \left(\sum_{j=1}^{n} y_j^p\right)^{1/p}$. We can now bound the solution to this optimization:

$$\begin{aligned} \frac{\|k_j\|_1 \|x_j\|_{1/m}}{\|k_j x_j\|_{1/(m+1)}} &\leq \frac{\|k_j\|_1}{\|k_j x_j\|_{1/(m+1)}} n^m \\ &\leq \frac{\|k_j\|_1}{\|k_j\|_{1/(m+1)}} n^m k^{1-m} \\ &\leq n^m k^{1-m}. \end{aligned}$$

The first step follows from upper bounding $x_j \leq 1$. The second step follows from lower bounding $x_j \geq k^{m-1}$. The third step follows from observing that $\|k_j\|_{1/(m+1)} \geq \|k_j\|_1$.

To see that this bound on the price of anarchy is asymptotically tight, let us consider the situation with n queues where $k_1 = k$ and $k_2 = \ldots = k_n = 1$. In this case,

$$\begin{aligned} \frac{\left(\sum_{j=1}^{n} k_j\right)\left(\sum_{j=1}^{n} k_j^{\frac{m-1}{m}}\right)^m}{\left(\sum_{j=1}^{n} k_j^{\frac{m}{m+1}}\right)^{m+1}} &= \frac{(n-1+k)(n-1+k^{\frac{m-1}{m}})^m}{(n-1+k^{\frac{m}{m+1}})^{m+1}} \\ &\geq \frac{(n-1)^{m+1} + k(n-1)^m}{(n-1+k^{m/(m+1)})^{m+1}} \\ &= \frac{1+\frac{k}{n-1}}{\left(1+\frac{k^{m/(m+1)}}{n-1}\right)^{m+1}}. \end{aligned}$$

Now, suppose $k, n \to \infty$ with $k^{m/(m+1)} \gg n$, then we have

$$\begin{aligned} \frac{1+\frac{k}{n-1}}{\left(1+\frac{k^{m/(m+1)}}{n-1}\right)^{m+1}} &\sim \frac{\frac{k}{n}}{\frac{k^m}{n^{m+1}}} \\ &= n^m k^{1-m}. \end{aligned}$$

■

VI. The case of arbitrary scheduling policies and general job sizes

In the previous section we derived bounds on the price of anarchy when the back-end scheduler performs SRPT and job sizes follow a Pareto distribution. In this section, we discuss generalizations of those results to both general job size distributions and to arbitrary policies.

Though generalizing the scheduling policy and generalizing the job size distribution seem different from one another, they both have the same effect – they change the form of $E[T_i]$. For example, Bansal [4] recently showed that the heavy-traffic growth rate of SRPT under Exponential job sizes is

$$E[T_i] = \theta\left(\frac{1}{(\mu_i - \lambda_i)\log(\mu_i/(\mu_i - \lambda_i))}\right). \tag{2}$$

Similar heavy-traffic results have recently been derived for policies such as Preemptive Shortest Job First (PSJF) [31], Foreground-Background scheduling (FB) [20], and multi-class priority queues [30] under a variety of job size distributions. The derivation of these results is an active area.

Our goal in this section is to provide results that can easily facilitate the calculation of the price of anarchy under arbitrary scheduling policies and job size distributions. That is, we would like to ensure that as new heavy-traffic results appear, price of anarchy results follow easily. We provide two such results. The first result illustrates that the price of anarchy is determined only by the polynomial term in the heavy-traffic growth rate and the second result characterizes a "simple" situation that can be analyzed to determine the price of anarchy in a very general setting. To illustrate the usefulness of these results, we will apply them to the case of SRPT scheduling with Exponential job sizes, (2).

Theorem 6. *Fix μ_i. Consider $C_i(\lambda_i)$ such that for all $\epsilon \in (0, m)$, there exists $\lambda_{i,\epsilon}$ such that for all λ_i with $\lambda_{i,\epsilon} < \lambda_i < \mu_i$*

$$\left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^{m-\epsilon} \le C_i(\lambda_i) \le \left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^{m}. \tag{3}$$

Then the price of anarchy in heavy-traffic under $C_i(\lambda_i)$ is the same as the price of anarchy in heavy-traffic under $\left(\frac{\mu_i}{\mu_i-\lambda_i}\right)^m$. In particular, the price of anarchy is $n^m k^{1-m}$.

Proof: Let us refer to the terms in (3) as A, B, and C respectively. Then, we will prove the result by showing that $E[T_A^{ne}] \le E[T_B^{ne}] \le E[T_C^{ne}]$ and $E[T_A^{opt}] \le E[T_B^{opt}] \le E[T_C^{opt}]$. The result follows because for any $\lambda_i < \mu_i$, as $\epsilon \to 0$, we have that $E[T_A^{ne}] \to E[T_C^{ne}]$ and $E[T_A^{opt}] \to E[T_C^{opt}]$.

First, note that it is almost immediate that $E[T_A^{opt}] \le E[T_B^{opt}] \le E[T_C^{opt}]$. In particular, $\lambda_i^{opt}(B)$ can be used as the arrival rates in A, and will give smaller mean response time than $E[T_B^{opt}]$ (for high enough Λ). Further, the optimal arrival rates will lead to an even smaller mean response time. A parallel argument shows $E[T_B^{opt}] \le E[T_C^{opt}]$.

Second, we need to argue that $E[T_A^{ne}] \le E[T_B^{ne}] \le E[T_C^{ne}]$. We will only argue the case of $E[T_A^{ne}] \le E[T_B^{ne}]$, since the remaining argument will be symmetric. Let us begin with the Nash assignment in B. Now, one at a time, we will switch the cost functions in B to match those in A. We start with the slowest queue. Consider what happens when we switch the cost function of a queue. If the arrival rate were to remain unchanged, the response time would drop (for large enough Λ). So, at Nash equilibrium, arrivals from other queues must shift to the queue that just changed. Thus, the overall mean response time drops. This happens with each change, so the resulting Nash assignment in case A is such that $E[T_A^{ne}] \le E[T_B^{ne}]$.

Finally, since the above ordering holds for all ϵ and the price of anarchy bound is continuous in m, we have that B has price of anarchy that matches C, as desired. ■

Notice that an immediate corollary of the above theorem is that SRPT has a price of anarchy of n in the case of Exponential job sizes.

Our next result characterizes price of anarchy by determining an easy to study set of systems that is guaranteed to contain the worst case price of anarchy. Thus, it provides a "simple" way to calculate the price of anarchy.

Theorem 7. *Consider a system with $C_i(\lambda_i) = f(1-\rho_i)$ for some fixed non-increasing function f where $f(x) - f(cx) \le f(y) - f(cy)$, for all $c < 1$, $y > x > 0$. Let S be such a system having n queues where the service rates of these queues are between μ and $k\mu$. Then, there exists a system S' with a queues of service rate $k\mu$ and b queues with service rate μ where $b/a \le n-1$ such that the price of anarchy for S is less than or equal to price of anarchy for S'.*

Theorem 7 shows that to calculate the price of anarchy, it suffices to study only systems having queues with *two* different service rates, $k\mu$ and μ. Notice that this theorem both provides a "simple" way to calculate the price of anarchy and provides an illustration of the tightness of the price of anarchy. In the case of SRPT and Pareto job sizes the this result led to the determination that the worst-case scenario was $\mu_1 = k\mu$ and $\mu_2 = \ldots = \mu_n = \mu$. It also applies easily in the case of SRPT under Exponential job sizes (2) where a simple calculation shows that the worst-case is again $\mu_1 = k\mu$ and $\mu_2 = \ldots = \mu_n = \mu$. It then follows quickly that the price of anarchy is n. The details of this argument as well as the proof of Theorem 7 are omitted due to space constraints.

VII. Concluding remarks

Server farms are now the dominant enterprise system architecture, and load balancing dispatchers are by far the most common design for such systems. However, load balancing designs are well-known to be inefficient in terms of the overall mean response time. In this paper, we have provided results characterizing the inefficiency of load balancing designs.

In particular, we have focused on the design of the back-end scheduler and shown that the inefficiency and performance of load balancing systems is highly dependent on the choice of the back-end scheduler. We showed that when the back-end scheduler is PS, as is common in traditional designs, the inefficiency of the system grows linearly with the number of back-end servers. In contrast, by switching the back-end scheduler to SRPT, the performance of the load balancing system can be greatly improved as long as the heterogeneity of server speeds is small. Further, the inefficiency of load balancing is less dependent on the number of servers, and more dependent on the heterogeneity of the server speeds.

These results provide interesting managerial-level insight into the design and management of server farms. In particular, the architecture of the server farm should be dependent on the back-end scheduler. If the back-end scheduler is PS, the ideal load balancing system consists of fewer, faster servers; whereas under SRPT, the ideal load balancing system consists of a large number of homogeneous servers. This insight begs the question of whether using heterogeneous back-end schedulers would be beneficial? Specifically, it may be beneficial to use SRPT on a large number of back-end servers with similar speeds and use PS on the back-end servers with very fast/small speeds. Studying this idea is a current topic of research.

The analysis in this paper is made possible by combining ideas from algorithmic game theory with recent heavy-traffic queueing results. Our results in Section VI provide a general technique for deriving price of anarchy results for each new heavy-traffic result that appears. However, one important open question that remains is how to exploit heavy-traffic queueing results in more general network structures.

REFERENCES

[1] E. Altman, U. Ayesta, and B. J. Prabhu. Optimal load balancing in processor sharing systems. In *Proc. of GameComm*, 2008.
[2] M. Arlitt and C. Williamson. Web server workload characterization: the search for invariants. In *Proc. of ACM Sigmetrics*, 1996.
[3] B. Avi-Itzhak, H. Levy, and D. Raz. A resource allocation fairness measure: properties and bounds. *Queueing Systems Theory and Applications*, 56(2):65–71, 2007.
[4] N. Bansal. On the average sojourn time under M/M/1/SRPT. *Oper. Res. Letters*, 22(2):195–200, 2005.
[5] N. Bansal and D. Gamarnik. Handling load with less stress. *Queueing Systems*, 54(1):45–54, 2006.
[6] O. Boxma and B. Zwart. Tails in scheduling. *Perf. Eval. Rev.*, 34(4):13–20, 2007.
[7] J. Cao, W. Cleveland, D. Lin, and D. Sun. Internet traffic tends *toward* poisson and independent as the load increases. Springer, New York, 2002.
[8] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *Trans. on Networking*, 5(6):835–846, 1997.
[9] D. G. Down and R. Wu. Multi-layered round robin routing for parallel servers. *Queueing Sys.*, 53(4):177–188, 2006.
[10] A. B. Downey. A parallel workload model and its implications for processor allocation. In *Proc. of High Performance Distributed Computing*, pages 112–123, August 1997.
[11] M. Harchol-Balter, B. Schroeder, M. Agrawal, and N. Bansal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(2), May 2003.
[12] J. M. Harrison. *Brownian motion and stochastic flow systems*. John Wiley and Sons, New York, USA, 1985.
[13] M. Haviv and T. Roughgarden. The price of anarchy in an exponential multi-server. *Oper. Res. Letters*, 35:421–426, 2007.
[14] M. Hu, J. Zhang, and J. Sadowsky. A size-aided opportunistic scheduling scheme in wireless networks. In *Globecom*, 2003.
[15] L. Kleinrock. *Queueing Systems*, volume II. Computer Applications. John Wiley & Sons, 1976.
[16] D. Lu, P. Dinda, Y. Qiao, and H. Sheng. Effects and implications of file size/service time correlation on web server scheduling policies. In *Proc. of IEEE MASCOTS*, 2005.
[17] D. Lu, H. Sheng, and P. Dinda. Size-based scheduling policies with inaccurate scheduling information. In *Proc. of IEEE MASCOTS*, 2004.
[18] R. Mangharam, M. Demirhan, R. Rajkumar, and D. Raychaudhuri. Size matters: Size-based scheduling for MPEG-4 over wireless channels. In *SPIE & ACM Proceedings in Multimedia Computing and Networking*, pages 110–122, 2004.
[19] N. Nissan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, New York, NY, USA, 2007.
[20] M. Nuyens and A. Wierman. The foreground-background queue: A survey. *Performance evaluation*, 65(3-4):286–307, 2008.
[21] M. Nuyens, A. Wierman, and B. Zwart. Preventing large sojourn times using SMART scheduling. *Oper. Res.*, 56(1):88–101, 2008.
[22] A. Pigou. *The Economics of Welfare*. Macmillan, 1920.
[23] M. Rawat and A. Kshemkalyani. SWIFT: Scheduling in web servers for fast response time. In *Symp. on Net. Comp. and App.*, 2003.
[24] T. Roughgarden. The price of anarchy is independent of the network topology. *J. Comp. Syst. Sci.*, 67(2):341–364, 2003.
[25] T. Roughgarden and E. Tardos. How bad is selfish routing. *J. ACM*, 49(2):236–259, 2002.
[26] L. E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:678–690, 1968.
[27] L. E. Schrage and L. W. Miller. The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*, 14:670–684, 1966.
[28] J. G. Wardrop. Some theoretical aspects of road traffic research. *Proc of Institute of Civil Engineers*, 1:325–378, 1952.
[29] A. Wierman. Fairness and classifications. *Perf. Eval. Rev.*, 34(4):4–12, 2007.
[30] A. Wierman. *Scheduling for today's computer systems: Bridging theory and practice*. PhD thesis, Carnegie Mellon University, 2007.
[31] A. Wierman, M. Harchol-Balter, and T. Osogami. Nearly insensitive bounds on SMART scheduling. In *Proc. of ACM Sigmetrics*, 2005.
[32] A. Wierman and M. Nuyens. Scheduling despite inexact job-size information. In *Proc. of ACM Sigmetrics*, 2008.
[33] R. J. Williams. Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse. *Queueing Systems Theory Appl.*, 30(1-2):27–88, 1998.
[34] R. Wu and D. G. Down. On the relative value of local scheduling versus routing in parallel server systems. In *Proc. of Conf. on Paralell and Dist. Sys.*, 2007.
[35] R. Wu and D. G. Down. Round robin scheduling of heterogeneous parallel servers in heavy traffic. In *European J. of Oper. Res.*, to appear.
[36] T. Wu and D. Starobinski. On the price of anarchy in unbounded delay networks. In *Proc. of Game Theory for Comm. and Networks*, 2006.
[37] S. Yashkov. Mathematical problems in the theory of shared-processor systems. *J. of Soviet Mathematics*, 58:101–147, 1992.
[38] J. Zhang, J. Dai, and B. Zwart. Diffusion limits of limited processor sharing queues. *submitted for publication*, 2007.

APPENDIX A
CALCULATING THE OPTIMAL ASSIGNMENTS

In this section we will calculate the optimal routing assignment in the case of SRPT scheduling and Pareto job sizes. The optimal dispatcher will use a routing scheme that minimizes the overall mean response time $E[T]$; thus the dispatcher needs to determine the λ_i^{opt} that solve the following optimization:

$$\begin{aligned} &\min \sum_{i=1}^{n} C_i(\lambda_i) \\ &\text{s.t.} \sum_{i=1}^{n} \lambda_i = \Lambda; \\ &\quad 0 \le \lambda_i < \mu_i, \quad 1 \le i \le n. \end{aligned}$$

We can solve this optimization explicitly to obtain the following lemmas, which are summarized in Table IV-B.

Lemma 8. *Consider* $C_i(\lambda_i) = \log\left(\frac{\mu_i}{\mu_i - \lambda_i}\right)$. *The optimal dispatcher uses*

$$\lambda_j^{opt} = \mu_j - \frac{\sum_{i=1}^{n} \mu_i - \Lambda}{n}.$$

Proof: First note that the optimal arrival rates must satisfy

$$\frac{d}{d\lambda_i} C_i(\lambda_i) = \frac{d}{d\lambda_j} C_j(\lambda_j)\ \forall i,j \in \{1,...,n\}. \tag{4}$$

This simplifies to

$$\lambda_i^{opt} - \mu_i = \lambda_j^{opt} - \mu_j,\ \forall i,j \in \{1,...,n\}. \tag{5}$$

Recall that the *gap* at Q_i, is $\gamma_i := \mu_i - \lambda_i$. So, the above equation gives us that the $\gamma_i^{opt} = \gamma_j^{opt}$ for all i, j. It follows that

$$\gamma_i^{opt} = \Gamma/n,$$

or equivalently

$$\lambda_j^{opt} = \mu_j - \frac{\sum_{i=1}^{n} \mu_i - \Lambda}{n}.$$

■

Notice the intuition to the above lemma: the total excess service capacity ($\sum_{i=1}^n \mu_i - \Lambda$) is divided evenly among the servers. In this next lemma, the total excess service capacity ($\sum_{i=1}^n \mu_i - \Lambda$) is not divided evenly among the servers anymore, instead it is divided in proportion to $\mu_i^{m/(m+1)}$.

Lemma 9. *Consider* $C_i(\lambda_i) = \left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m$. *The optimal dispatcher uses*

$$\lambda_j^{opt} = \mu_j - \left(\frac{\mu_j^{m/(m+1)}}{\sum_{i=1}^n \mu_i^{m/(m+1)}}\right)\left(\sum_{i=1}^n \mu_i - \Lambda\right).$$

Proof: We can again find the socially optimal arrival rates by solving (4). This gives

$$\frac{m}{\mu_i}\left(\frac{\mu_i}{\mu_i - \lambda_i^{opt}}\right)^{m+1} = \frac{m}{\mu_j}\left(\frac{\mu_j}{\mu_j - \lambda_j^{opt}}\right)^{m+1}, \quad (6)$$

from which we obtain

$$\frac{\gamma_i^{opt}}{\gamma_j^{opt}} = \left(\frac{\mu_i}{\mu_j}\right)^{m/(m+1)}.$$

So, we can write

$$\sum_{i=1}^n \gamma_i^{opt} = \sum_{i=1}^n \gamma_j^{opt}\left(\frac{\mu_i}{\mu_j}\right)^{m/(m+1)},$$

which gives

$$\gamma_j^{opt} = \left(\frac{\mu_j^{m/(m+1)}}{\sum_{i=1}^n \mu_i^{m/(m+1)}}\right)\Gamma.$$

Equivalently, we have

$$\lambda_j^{opt} = \mu_j - \left(\frac{\mu_j^{m/(m+1)}}{\sum_{i=1}^n \mu_i^{m/(m+1)}}\right)\left(\sum_{i=1}^n \mu_i - \Lambda\right).$$

Note that when $m = 1$ we get the optimal arrival rates for PS. ■

APPENDIX B
CALCULATING THE NASH ASSIGNMENT

In this section we will calculate the Nash assignment in the case of SRPT scheduling and Pareto job sizes. We know that in heavy-traffic all queues are used and thus the arrival rates must satisfy

$$E[T_i] = E[T_j],\ \forall i,j \in \{1,...,n\}.$$

From this condition, it is possible to derive the arrival rates explicitly, and we attain the results summarized in Table IV-B.

Lemma 10. *Consider* $C_i(\lambda_i) = \log\left(\frac{\mu_i}{\mu_i - \lambda_i}\right)$. *Then,* λ_j^{ne} *satisfies*

$$1 - \frac{\lambda_j^{ne}}{\mu_j} = \left(1 - \frac{\lambda_1^{ne}}{\mu_1}\right)^{\mu_j/\mu_1},$$

where λ_1^{ne} *is the solution to*

$$\sum_{i=1}^n \mu_i - \Lambda = \sum_{i=1}^n \mu_i\left(1 - \frac{\lambda_1^{ne}}{\mu_1}\right)^{\mu_i/\mu_1}.$$

Proof: At a Nash assignment, all jobs must have the same expected response time, i.e., if $\lambda_1^{ne}, ..., \lambda_n^{ne}$ represent the arrival rates at a Nash assignment, then for all $i,j \in \{1,...,n\}$

$$\frac{1}{\mu_i}\log\left(\frac{\mu_i}{\mu_i - \lambda_i^{ne}}\right) = \frac{1}{\mu_j}\log\left(\frac{\mu_j}{\mu_j - \lambda_j^{ne}}\right). \quad (7)$$

From here, we can calculate λ_i^{ne} explicitly. From the equation above, it follows that

$$\left(\frac{\mu_i}{\gamma_i^{ne}}\right)^{1/\mu_i} = \left(\frac{\mu_j}{\gamma_j^{ne}}\right)^{1/\mu_j}. \quad (8)$$

Summing both sides gives

$$\sum_{i=1}^n (\gamma_i^{ne})^{1/\mu_i} = \sum_{i=1}^n \mu_i^{1/\mu_i}\left(\frac{\gamma_j^{ne}}{\mu_j}\right)^{1/\mu_j},$$

from which it follows that

$$(\gamma_j^{ne})^{1/\mu_j} = \left(\frac{\mu_j^{1/\mu_j}}{\sum_{i=1}^n \mu_i^{1/\mu_i}}\right)\sum_{i=1}^n (\gamma_i^{ne})^{1/\mu_i}. \quad (9)$$

Combining (8) and (9) gives

$$\gamma_j^{ne} = \mu_j\left(\frac{\gamma_1^{ne}}{\mu_1}\right)^{\mu_j/\mu_1}$$
$$\Gamma = \sum_{i=1}^n \mu_i\left(\frac{\gamma_1^{ne}}{\mu_1}\right)^{\mu_i/\mu_1},$$

which is equivalent to the equation in the statement of the lemma. ■

Note that, though the form of λ_i^{ne} in the above lemma is implicit, it can be solved easily in many special cases. For instance, when $\mu_i = \mu_j$ for all i,j, the Nash assignment is the same as the optimal assignment.

The λ_i^{ne} in the next lemma are explicit. In fact, in this case λ_j^{ne} has nearly the same form as λ_j^{opt} in Lemma 9.

Lemma 11. *Consider* $C_i(\lambda_i) = \left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m$. *Then*

$$\lambda_j^{ne} = \mu_j - \left(\frac{\mu_j^{(m-1)/m}}{\sum_{i=1}^n \mu_i^{(m-1)/m}}\right)\left(\sum_{i=1}^n \mu_i - \Lambda\right).$$

Proof: The Nash condition gives us that

$$\frac{1}{\mu_i}\left(\frac{\mu_i}{\mu_i - \lambda_i}\right)^m = \frac{1}{\mu_j}\left(\frac{\mu_j}{\mu_j - \lambda_j}\right)^m,\ \forall i,j \in \{1,...,n\}. \quad (10)$$

Noting that this is parallel to (6) except that $m+1$ is now changed to m, we can immediately write

$$\gamma_j^{ne} = \left(\frac{\mu_j^{(m-1)/m}}{\sum_{i=1}^n \mu_i^{(m-1)/m}}\right)\Gamma,$$

which is equivalent to the statement of the lemma. ■