Energy-Aware Scheduling on Heterogeneous Processors

Osman T. Akgun, Douglas G. Down, *Senior Member, IEEE,* and Rhonda Righter

Abstract

We study a multi-server system where servers are heterogeneous in terms of both their speeds and their usage (energy) costs. We show that the optimal control policy is threshold type, and can be implemented in a decentralized manner through an individually optimal policy. When there are no arrivals we also derive explicit expressions for the thresholds. We consider both the case with reassignment, where jobs can be moved from one server to another at any time, and the case where reassignment is not possible. When reassignment is possible, determining which server is preferred (i.e., the one that is used even if there are very few jobs waiting) is surprisingly difficult. With arrivals, for both the reassignment and no-reassignment cases, determining server preference is also difficult, and depends on the arrival rate.

I. Introduction

Energy consumption has been growing exponentially in computers and computer centers [1], embedded systems, portable devices, etc., and it is also a critical concern in battery-operated devices such as sensors. Energy cost has become the critical cost factor for server farms, and there is also increasing concern over the heat and carbon emissions generated [2]. No longer is faster necessarily better; both speed and energy consumption must be considered in the operation of these systems, and often faster servers require more energy. Hence the emphasis is on providing good service while maintaining energy efficiency. We study a multi-server system where servers are heterogeneous in terms of both their speeds and their usage costs.

Manuscript received May 12, 2013.

O. T. Akgun and R. Righter are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA, 94720 USA (e-mail: akguno@ieor.berkeley.edu, rrighter@ieor.berkeley.edu).

D. G. Down is with the Department of Computing and Software, McMaster University, Hamilton, ON, L8S 4L7, Canada (e-mail: downd@mcmaster.ca).

Including usage costs is also a means of improving fairness in systems where the servers are humans, such as call centers [3]. Applying a higher charge to a faster call center agent can help balance workload in a fairer manner across agents with heterogeneous speeds.

The heterogeneous server problem has been studied in the literature in terms of both admission control and scheduling control. Admission control has received a lot of attention in the operations management community (see, e.g., [4], [5], [6], [7], and [8]).

We consider optimal scheduling of homogeneous jobs on heterogeneous servers, i.e., deciding whether or not to immediately assign a waiting job to an available server in order to minimize mean total holding costs and server usage costs. All jobs in the system (including those receiving service) incur a holding cost and the jobs using the servers incur a usage cost (different for each server) per unit time. Usage costs are not incurred when servers are idle. Service times are exponentially distributed, and customers arrive according to a Poisson process. The case with only holding costs and different server speeds (the slow-server problem) has been extensively studied. Weiss and Pinedo [9] considered the slow-server problem where reassignment is possible, that is, the jobs can be reassigned to other servers at any given time even when they are in service. They showed that the optimal policy is to use all the servers if there are more jobs than servers, and to otherwise serve all the jobs on the fastest servers. More work has been done on the harder problem in which jobs cannot be reassigned. Agrawala et al. [10] showed that with no arrivals, the optimal policy is threshold type (with a threshold on the number of jobs determining whether a server is used), and the threshold for a given server is independent of the state of the slower servers. With arrivals and only two servers, the optimal policy is again threshold type. Lin and Kumar [11] and Koole [12] showed this result using dynamic programming, Walrand [13] showed it with sample path arguments, Xu [14] showed it by comparing the individually optimal and socially optimal policies, and Stockbridge [15] showed it using a martingale approach. Viniotis and Ephremides [16] extended the results to fairly general arrival processes and service-time distributions. When there are more than two servers and no reassignment, the problem turns out to be much more difficult. Rykov [17] used dynamic programming to show that the optimal policy is a state-dependent threshold policy but de Vèricourt and Zhou [18] showed that Rykov's proofs are incomplete. A linear programming approach was used by Luh and Viniotis [19] to show the optimal policy is threshold type, but their proofs are opaque. Rosberg and Makowski [20] showed that for sufficiently small arrival rates, the optimal policy is the same as the case with no arrivals. For more general arrival rates Weber [21] showed that, in contrast to the no-arrival case, if there exists an optimal threshold policy, the optimal threshold for using a server depends on the state of the slower servers. Kim, Ahn and Righter [22] showed that a threshold policy for primary jobs is optimal when there is an infinite number of secondary jobs, and these jobs are assigned to servers that are not used for primary jobs.

In the literature the problem with server usage costs also usually includes server setup (on/off) costs. For two homogeneous servers, Bell [23] proved that the optimal policy is a hysteresis policy with two thresholds for turning the server off or on. For more than two servers or for heterogeneous servers, the optimality of a hysteresis policy is generally assumed [24], [25], [26]. There has also been some work in the area of dynamically changing the speeds (speed scaling) of individual servers in the presence of speed-dependent usage costs [27], [28]. We assume that particular servers have fixed speeds and usage costs, and there are no setup costs.

Without setup costs the problem is still difficult. Weiss and Pinedo [9] studied minimizing the server usage costs in a system without arrivals and where reassignment is possible, but the policies they consider exclude threshold-type policies. Rykov and Efrosinin [29] studied the system without reassignment and with both server usage costs and holding costs, but these costs are agreeable in the sense that faster servers have smaller usage costs. This assumption simplifies the problem considerably, because the order in which servers are preferred becomes trivial.

As mentioned earlier, Xu [14], building on ideas of Hassin [30], used the relation between social optimality and individual optimality to show the earlier known threshold result for two-server scheduling control. Righter and Xu [31] also used this approach for multiple servers and no arrivals when the service times have an IFR (Increasing Failure Rate) distribution. We also use this approach. In particular, we show the individually optimal policy is a threshold policy, which is fairly easy, and then show, under appropriate priorities for the individually optimal policy, that it is also socially optimal. The first part is easy, because under an individually optimal policy, jobs are given a priority order and then offered available servers according to their priority order. If, for a given state, job t (the tth job in priority order) chooses to use server j, then, whenever there are at least t jobs in the system and the state is otherwise the same, job t will use server j. For the individually optimal policy to also be socially optimal, we must assign priorities so that the lowest priority job has no impact on the other jobs (see, e.g., Hassin [30]). With arrivals, this will require a last-come first-priority rule, with preemption (LCFP-P). Then the job at the back of the queue imposes no externality, and a backwards induction argument, involving a change of policy for the lowest priority customer, gives the result. Our approach, of using an appropriate individually optimal policy to derive the socially optimal policy, gives more intuition into both the structure of the optimal policy and the particular values of the thresholds than a standard dynamic programming approach, and we can use the individually optimal policy to implement the socially optimal policy in a decentralized fashion. Note, however, that once we have shown that a threshold policy is socially optimal, and we have the thresholds, we can also implement the policy in a centralized fashion, by assigning an *arbitrary* job to a server if there are at least the threshold number of jobs in the system. For example, jobs could be assigned in FCFS (first-come first-served) order.

Showing that the optimal policy is threshold type is similar to the problem with only holding costs (the slow-server problem) in most of the cases we consider. The difficulty caused by the addition of the server usage costs emerges when we try to calculate the values for optimal thresholds, and even in determining the server preference relation. When there are no usage costs, it is simple to show that faster servers are always preferred. For the model with usage costs, even with no arrivals and reassignment, the server preference relations are complicated expressions that depend on holding costs, usage costs and all the server rates. Moreover, when we allow arrivals and consider only two servers, we show, somewhat surprisingly, that the choice of preferred server depends on the arrival rate. Indeed, for a system with a finite number of arrivals, the preferred server can change with the state of the system.

In Section II, we study the clearing system (no arrivals) and in Section III we consider arrivals. In both sections, we analyze two different problems, one in which jobs can be reassigned from one server to another or removed from a server and put in the queue at no cost (also known as job migration), and one in which, after being assigned to a server, a job cannot be removed from that server. An outline of our results is as follows.

- For the clearing system both with and without reassignment, where jobs are initially assigned arbitrary priorities, the individually optimal policy has the following properties.
  - A job's individually optimal decision and cost are independent of the lower priority jobs (Corollaries 2 and 8).
  - We give explicit simple formulas for v_i, the cost under the individually optimal policy for job i, that involve combining multiple servers into a single server with the sum of the costs and service rates (Lemmas 4 and 11).
  - The individually optimal policy is threshold type, and we give explicit formulas for server preferences and thresholds (Lemma 3 and Theorems 5 and 12).
- The socially optimal policy for the clearing system is the same as the individually optimal policy, both with and without reassignment (Theorems 13 and 17, using Corollaries 2 and 8 above).
- With arrivals, under LCFP-P, the individually optimal policy is threshold type (Theorems 19 and 21).

- The socially optimal policy with arrivals is the same as the individually optimal policy with reassignment, and, for two servers, without reassignment (Theorems 20 and 22).
- For two servers and arrivals, we give properties of the optimal policy, and show how it depends on the arrival rate. For example, cheaper but slower servers will be preferred at smaller arrival rates (Theorems 26, 28, 32) and the threshold for using the less preferred server is decreasing in the arrival rate (Corollaries 29 and 33). We also give an algorithm to determine the threshold in the problem with reassignment.

II. The Clearing System

We first consider a multi-server queueing system with s parallel servers, where at time 0 there are n homogeneous jobs that need to be processed, and no new arrivals (i.e., a clearing system). The service times are independently and exponentially distributed with rate μ_j on server j, $j = 1, ..., s$. Whenever a job is processed on server j, it incurs a cost with rate β_j. While in the system, each job incurs a holding cost with rate h. Our objective is to minimize the total expected cost (or equivalently, the per job average cost). We will show that the socially optimal policy is threshold type. This means the servers can be ordered in preference order such that, for any state where servers i and j are available, and $i < j$, the socially optimal policy will assign a job to i before assigning a job to j (i.e., it will not assign a job to j and leave i idle). Moreover, given this server preference order, the socially optimal policy uses thresholds $t_1 \leq t_2 \leq \ldots \leq t_s$ such that if server i is the lowest indexed server available, a job will be assigned to server i if and only if the number of waiting jobs is at least t_i. Note that for social optimality it does not matter which job is assigned. Indeed, because of the exponential distribution and because jobs are indistinguishable, the socially optimal policy can replace jobs that are being served with waiting jobs at any time. Also note that the threshold for server j does not depend on the states of the less preferred (higher indexed) servers.

A. *The Individually Optimal Policy*

We assume that waiting (unassigned) jobs are given arbitrary priorities, and each job attempts to minimize its own holding cost and server usage cost, subject to the priorities. In the problem with reassignment, we assume (without loss of generality) that all jobs are initially waiting; in the problem without reassignment some jobs may already be assigned to servers. Idle servers are offered to the unassigned job with the highest priority, job 1; if that job chooses to wait for one of the busy servers, or if there is more than one idle server, then the job with the second highest priority, job 2, will be offered the remaining idle servers. The process continues until all servers are busy or all the jobs have been considered for assignment, and then is repeated at the next service completion. Because servers are offered to jobs in priority order, it is immediate that a threshold policy will be individually optimal, i.e., for i large enough, the ith priority job will choose to use a less desirable server that higher priority jobs have rejected, say server j, rather than wait for a more desirable server. More explicitly, let us order the servers in decreasing order of desirability (so that if a job is offered both servers k and j, $k < j$, it will choose to use k). Then for each server j there is a threshold, t_j, such that if servers $1, ..., j-1$ are busy and j is idle, server j will be used as long as there are at least $t_j = i$ jobs in the system. In principle this threshold could depend on the states of less desirable servers $k > j$ (especially with no reassignment), but below we obtain explicit values for the optimal thresholds and show that the threshold for a server is independent of the states of less desirable servers.

Note that for the slow-server problem (with no usage costs), the problem with reassignment is much easier than the the problem without reassignment. Indeed, it is easy to show that the ith job should use the ith server as long as there are at least i servers, so the fastest $\min(n, s)$ servers will always be used. However, with heterogeneous usage costs, both problems are more difficult, and, surprisingly, the problem without reassignment is easier (and, of course, will lead to higher total costs). Hence, we start with this case.

1) Problem without reassignment: For now we assume that if a job chooses to use a server, it will stay with that server until service completion and no job can preempt another job on a server. For instance, suppose we have two servers and at time 0 server 1 is busy, server 2 is idle and three jobs are waiting. Then server 2 will be offered to job 1. Let us assume that job 1 rejects server 2 and prefers to wait for server 1 (so it will never use server 2). Then server 2 will be offered to job 2. Let us assume that job 2 accepts the server and begins service. Then job 3 becomes job 2 and the next assignment will occur when there is a departure. If the departure is from server 1, then job 1 will take server 1. If the departure is from server 2, then job 1 will reject it and job 2 (formerly job 3) will make the same decision as the previous job 2, and will accept server 2. The remaining job will then wait for server 1.

We first give two properties that an individually optimal policy must satisfy, and then we derive a surprisingly simple expression for the expected cost for each job for any priority policy that satisfies those properties. Using this we derive optimal thresholds and the optimal expected costs for the jobs based on their priorities. We call Lemma 1 the *monotonicity* result, following Coffman et al. [32] for the slow-server problem.

Lemma 1: If a server is rejected by a job under the individually optimal policy, it will never be used by that job in the future.

Proof: The result is intuitive for exponential service times, because after rejecting a server, a job's position only improves: it will later be offered a better server, or it will move up in the queue because a higher priority customer starts service on a better server. We will consider the ith job in the queue (the unassigned job with the ith highest priority), and show the result by induction on i. First consider $i = 1$, and suppose there is $n = 1$ waiting job. The result is immediate: If the first (only) job in queue rejects the idle servers, then it has to be served by one of the busy servers, because waiting and then using one of the originally available servers will incur additional holding cost without reducing operating cost. This remains true for $i = 1$ and $n \geq 1$, because lower priority jobs will have no effect on job 1. Now suppose the result is true for $i-1$ and suppose $n = i$. Let I_i denote the set of idle servers offered to the ith job. By definition of the priority policy, the servers in I_i were also offered to jobs $1, 2, ..., i-1$ and were rejected, and by induction, will never be used by those jobs. Suppose the ith job also rejects the servers in I_i, and prefers to wait for one of the other servers. It will keep waiting as the ith job in queue until one of the following cases happens. Case I: a departure occurs at one of the servers that is not in I_i and this server is offered to the ith job and is accepted. Then job i will stay with that server until departure, and the result will be true. Case II: a departure occurs at one of the servers that is not in I_i and this server is accepted by one of the jobs $1, 2, ..., i-1$. Then the ith job becomes the $i-1$st job and it still will not use the servers in I_i because job $i-1$ rejected those servers at the initial decision epoch, so the result follows by the induction hypothesis. Case III: a departure occurs at one of the servers that is not in I_i and jobs $1, ..., i$ all reject the server, so I_i is augmented by this server and we continue. Job i will not use a server in I_i in the meantime because it will incur more holding cost for waiting and using the server than it would have for using it in the first place. The argument also holds for any $n \geq i$, and we are done. ∎

Note that Lemma 1 implies that we do not have a "game" situation, because a job's decision has no effect on higher priority jobs, so it can take their decisions as fixed.

Corollary 2: A job's individually optimal decision and cost are independent of the lower priority jobs.

Because $\frac{h+\beta_j}{\mu_j}$ is the expected cost of using server j, we also have the following lemma.

Lemma 3: For all j, k, server j is preferred to server k if $\frac{h+\beta_j}{\mu_j} \leq \frac{h+\beta_k}{\mu_k}$.

Let us label the servers according to the index of Lemma 3, $\frac{h+\beta_j}{\mu_j} \leq \frac{h+\beta_{j+1}}{\mu_{j+1}}$, $j = 1, ..., s-1$. Note that in the classic slow-server problem ($\beta_j = 0$ for all $1 \leq j \leq s$), the labeling is in decreasing order of server speed. Throughout we use increasing, etc., in the nonstrict sense.

Let us define an arbitrary (not necessarily individually optimal) priority policy that satisfies Lemmas 1 and 3 as follows. Given all servers are busy initially, let S_i denote the set of servers that would not be rejected by job i if one of those were next to become available (i.e., either acceptable to i or taken by $1, ..., i-1$). Note that by Lemma 3, we have $S_i = \{1, 2, ..., \sigma_i\}$ for some $\sigma_i \in S = \{1, ..., s\}$, and by Lemma 1 this set is fixed. We also have that σ_i is increasing in i, i.e., $S_i \subseteq S_{i+1}$, because servers are offered to jobs according to the priority order. Given any collection of sets $\underline{S} = \{S_1, S_2, ...\}$, such that $S_i = \{1, 2, ..., \sigma_i\}$ and σ_i is increasing in i, the corresponding individual priority policy is determined as follows. Suppose server j is the lowest indexed available server. Then it will be accepted by job i where i is such that $\sigma_{i-1} < j \leq \sigma_i$, $i \leq n$. Jobs $i+1, i+2, ..., n$ then become jobs $i, i+1, ..., n-1$ and the process is repeated with the next available server with the smallest index. Thus, the σ_i's determine a threshold policy.

Given $\underline{S} = \{S_1, S_2, ...\}$, let $v_i(\underline{S})$ be the individual expected cost of job i given all jobs use the sets in $\underline{S}$ for their decisions as described above, and given all servers are initially busy, or equivalently all servers in S_i are busy (note that $v_i(\underline{S})$ does not depend on $\sigma_{i+1}, ..., \sigma_n$ because we assume the policy satisfies Lemma 1). Also, let $M_j = \sum_{k=1}^{j} \mu_k$, $B_j = \sum_{k=1}^{j} \beta_k$. Then we have the following lemma.

Lemma 4: For a priority policy defined by $\underline{S} = \{S_1, S_2, ...\}$ such that $S_i = \{1, 2, ..., \sigma_i\}$ and σ_i is increasing in i,

$$v_i(\underline{S}) = \frac{(i+\sigma_i)h + B_{\sigma_i}}{M_{\sigma_i}} =: v_i(\sigma_i). \tag{1}$$

Proof: The proof is by induction on i. For $i = 1$, we show the result by conditioning on the next service completion of servers in S_1.

$$\begin{aligned} v_1(\underline{S}) &= \frac{h + \sum_{k=1}^{\sigma_1} \mu_k \frac{h+\beta_k}{\mu_k}}{M_{\sigma_1}} \\ &= \frac{(1+\sigma_1)h + B_{\sigma_1}}{M_{\sigma_1}} \end{aligned} \tag{2}$$

where the first equality follows because Lemma 1 is satisfied, i.e., the first job will never use servers $k > \sigma_1$. Suppose the result is true for $i-1$ and consider the ith job. Again by conditioning on the next service completion of servers in S_i, and using the induction hypothesis, we have, after some algebra,

$$\begin{aligned} v_i(\underline{S}) &= \frac{h + \sum_{k=1}^{\sigma_{i-1}} \mu_k v_{i-1}(\underline{S}) + \sum_{k=\sigma_{i-1}+1}^{\sigma_i} \mu_k \frac{h+\beta_k}{\mu_k}}{M_{\sigma_i}} \\ &= \frac{(i+\sigma_i)h + B_{\sigma_i}}{M_{\sigma_i}}. \end{aligned} \tag{3}$$

■

Note that Lemma 4 gives us an interesting interpretation of the expected costs for individual jobs. Consider job i, and suppose acceptable servers, $1, 2, ..., \sigma_i$, are all busy, so job i is the $i + \sigma_i$th job (including those on servers) in the system consisting of only acceptable servers. Then its expected cost is the same as if it were the $i+\sigma_i$th job in the system (i.e., the $i+\sigma_i-1$st waiting job) for a single-server system where the server usage cost is the sum of the costs for the first σ_i servers, and the speed is the sum of the speeds of the first σ_i servers. That is, the set of acceptable servers can effectively be collapsed into a single "super server." Note also that Lemma 4 holds more generally. In particular, we need not require servers to be ordered according to $\frac{h+\beta_j}{\mu_j}$, as long as $S_i \subseteq S_{i+1}$ and rejected servers are never used.

Lemma 1 tells us that the individually optimal policy is a simple threshold policy, and server j's threshold is $t_j = \min\{i : \sigma_i^* \geq j\}$, where σ_i^* is the individually optimal value of σ_i. Note also that a completely non-idling policy is also a threshold policy with all threshold values being equal to 1 (and $S_i \equiv S_1 = S$). We now derive explicit optimal threshold values.

Theorem 5: Without reassignment or arrivals, the individually optimal policy is threshold type, i.e., if the jth server is the available server with the lowest index, it will be used by job t_j (if $n \geq t_j$), where

$$t_j = \max\left\{t_{j-1}, \left\lceil C_j + \frac{M_{j-1}}{\mu_j} - (j-1)\right\rceil\right\},\ j = 2, ..., s, \tag{4}$$

$C_j = \frac{\beta_j M_{j-1} - \mu_j B_{j-1}}{h\mu_j}$, $t_1 = 1$, and t_j is increasing in j.

Proof: We use induction on j. It is easy to see that idling server 1 is not optimal, so $t_1 = 1$. Suppose (4) is true for $j-1$, so $\sigma_{t_{j-1}}^* \geq j-1$, and suppose server j is the lowest indexed available server and it is offered to job i. If $i < t_{j-1}$ then, from the induction hypothesis, $\sigma_i^* < j-1$, and job i will reject server j. Therefore suppose $i \geq t_{j-1}$, so $\sigma_i^* \geq j-1$. If job i rejects server j and waits for one of the servers $1, ..., j-1$ to be available, then its expected cost will be $v_i(j-1) = \frac{(i+j-1)h+B_{j-1}}{M_{j-1}}$ by Lemma 4. If it uses the jth server instead, then its expected cost will be $\frac{h+\beta_j}{\mu_j}$. Therefore $t_j = i$ for

$$i = \min\left\{k : \frac{h+\beta_j}{\mu_j} \leq \frac{(k+j-1)h + B_{j-1}}{M_{j-1}}, \text{ and } k \geq t_{j-1}\right\}. \tag{5}$$

Some algebra yields that the above is equivalent to (4) and the proof is complete. ■

When $\beta_j = 0$ (so $C_j = 0$), for all $1 \leq j \leq s$, Theorem 5 reduces to the classical slow-server result. Lemma 4 and Theorem 5 give us the following expressions for the individually optimal costs for each job, v_i. Of course the total cost will be $\sum_{i=1}^n v_i$.

Corollary 6: Under the individually optimal policy, $\sigma_i^* = j$ for $t_j \leq i < t_{j+1}$ and

$$v_i := v_i(\sigma_i^*) = \frac{(i+j)h + B_j}{M_j}, \text{ for } t_j \leq i < t_{j+1}. \tag{6}$$

Note that jobs can determine the optimal policy recursively in a decentralized fashion. Given σ^*_{i-1} and the lowest indexed available server $j > \sigma^*_{i-1}$, job i needs to compare only the super server generated by σ^*_{i-1} and server j, i.e., it compares $v_i(\sigma^*_{i-1})$ and $\frac{h+\beta_j}{\mu_j}$.

2) Problem with reassignment: Now jobs that are already using servers can be reassigned before completing service, so we can think of them as all being unassigned. We therefore assume a priority ordering on *all* jobs. At any time the highest priority job can choose any server, and the ith priority job can choose any server not being used by jobs $1, ..., i-1$. Therefore, lower priority jobs have no effect on higher priority jobs. The same argument as for the no-reassignment case also gives us monotonicity.

Lemma 7: If a server is rejected by a job, it will never be used by that job in the future.

Corollary 8: A job's individually optimal decision and cost is independent of the lower priority jobs.

The following is immediate by considering the server choice for the highest priority job.

Lemma 9: The most preferred server (the one that will be used if any jobs are present) is the one with the smallest value of $\frac{h+\beta_j}{\mu_j}$.

One might think that the preference index for server j, $j > 1$, will also be $\frac{h+\beta_j}{\mu_j}$, but this is only guaranteed if servers with smaller index are also faster. In particular, the preference ordering of less preferred servers depends on the parameters of the most preferred server, as shown in the next example.

Example 10: Suppose there are two jobs and three servers with $(h, \beta_1, \beta_2, \beta_3, \mu_1, \mu_2, \mu_3) = (2, 2, 5, 2, 2, 2, 1)$. Therefore $\frac{h+\beta_1}{\mu_1} < \frac{h+\beta_2}{\mu_2} < \frac{h+\beta_3}{\mu_3}$ and $v_1 = \min_j \frac{h+\beta_j}{\mu_j} = \frac{h+\beta_1}{\mu_1} = 2$.

Now consider the second job. If it waits for server 1, its expected cost will be $\frac{2h+\beta_1}{\mu_1}$. If it uses server 2, its expected cost will be $\frac{h+\beta_2}{\mu_1+\mu_2} + \frac{\mu_1}{\mu_1+\mu_2} v_1$. The first term is the cost incurred until the first service completion occurs, and the second term is the additional cost incurred if the service completion is at server 1. (The job can be reassigned, so if server 1 finishes first, the job will now be job 1 and will use server 1.) Plugging in v_1 yields $\frac{2h+\beta_1+\beta_2}{\mu_1+\mu_2}$ for the cost for job 2 if it uses server 2. Similarly if job 2 starts service on server 3, its expected cost will be $\frac{2h+\beta_1+\beta_3}{\mu_1+\mu_3}$. Hence,

$$\begin{aligned} v_2 &= \min\left\{\frac{2h+\beta_1}{\mu_1}, \frac{2h+\beta_1+\beta_2}{\mu_1+\mu_2}, \frac{2h+\beta_1+\beta_3}{\mu_1+\mu_3}\right\} \\ &= \min\left\{3, \frac{11}{4}, \frac{8}{3}\right\} = \frac{8}{3}, \end{aligned} \tag{7}$$

and job 2 will prefer server 3 to server 2 even though, if there were no server 1, server 2 would be preferred to server 3.

Let us again define an arbitrary (not necessarily individually optimal) priority policy that satisfies Lemmas 7 and 9 as follows. Let S_i denote the set of servers that would not be rejected by job i if one of those servers were next to become available (i.e., either acceptable to i or taken by any of jobs $1, ..., i-1$). We have that $S_{i-1} \subseteq S_i$ for all $i > 1$, i.e., $|S_i|$ is increasing in i, because servers are offered to jobs according to the priority order. We also assume without loss of generality that $|S_i \setminus S_{i-1}| \in \{0, 1\}$, i.e., if two servers are both acceptable and offered to job i (so they are not in S_{i-1}), then job i always chooses a particular one of them.

For $\underline{S} = \{S_1, S_2, ...\}$, an arbitrary collection of sets satisfying the conditions above, let $v_i(\underline{S})$ be the individual expected cost of job i given all jobs use the sets in $\underline{S}$ for their decisions. Also let $M(S_i) = \sum_{k \in S_i} \mu_k$, $B(S_i) = \sum_{k \in S_i} \beta_k$.

Lemma 11: For all $\underline{S} = \{S_1, S_2, ...\}$ such that $S_{i-1} \subseteq S_i$ and $|S_i \setminus S_{i-1}| \in \{0, 1\}$ for all $i \geq 1$,

$$v_i(\underline{S}) = \frac{ih + B(S_i)}{M(S_i)} =: v_i(S_i). \tag{8}$$

Proof: The proof is by induction on i. For $i = 1$, the result follows by Lemma 9, i.e., $S_1 = \{1\}$ and $v_1(\underline{S}) = \frac{h+\beta_1}{\mu_1}$. Suppose the result is true for $i - 1$ and consider the ith job. We look at the following two cases. First, suppose $S_{i-1} \neq S_i$. In this case job i starts taking service on server $S_i \setminus S_{i-1}$ (which is uniquely defined by our assumption that $|S_i \setminus S_{i-1}| \in \{0, 1\}$). By conditioning on the next service completion of servers in S_i and using the induction hypothesis, we have, after some algebra for the second equality,

$$\begin{aligned} v_i(\underline{S}) &= \frac{h + \beta_{S_i \setminus S_{i-1}} + \sum_{k \in S_{i-1}} \mu_k v_{i-1}(\underline{S})}{M(S_i)} \\ &= \frac{ih + B(S_i)}{M(S_i)}. \end{aligned} \tag{9}$$

When $S_{i-1} = S_i$, $v_i(\underline{S}) = h/M(S_i) + v_{i-1}(\underline{S}) = (ih + B(S_i))/M(S_i)$. ■

Note that, from Lemmas 4 and 11, for the same sets S_i that are acceptable to jobs $1, ..., i$, when all the servers in S_i are busy, the individual expected cost for the ith priority job, counting higher priority jobs in queue *and* higher priority jobs on servers in S_i, is the same for both models with and without reassignment. This is a bit surprising; one might think that the individual expected cost would be lower with reassignment. Consider an example with two servers that are both acceptable to any waiting jobs, and suppose both servers are busy and one job is waiting. If server 1 finishes first, then, in the problem without reassignment the waiting job will take that server and use it until it finishes, whereas, in the problem with reassignment the job on server 2 moves to server 1, and the waiting job takes server 2 (so it is worse off with reassignment if server 1 finishes first). On the other hand, if server 2 finishes first, the waiting job will take server 2 in both cases, but with reassignment it has the possibility of moving to server 1 (so the waiting job is better off with reassignment if server 2 finishes first). Note that the individual expected cost for using a server is, however, lower with reassignment.

Let S_i^* denote the individually optimal values of S_i. Let Π^* denote the permutation of the servers according to the optimal preference, so $S_i^* = \{\Pi^*(1), ..., \Pi^*(j)\} =: \Pi_j^*$ for some $1 \le j \le s$. Let $M(\Pi_j^*) = \sum_{k\in\Pi_j^*} \mu_k$, $B(\Pi_j^*) = \sum_{k\in\Pi_j^*} \beta_k$, and $C^k(\Pi_{j-1}^*) = \frac{\beta_k M(\Pi_{j-1}^*) - \mu_k B(\Pi_{j-1}^*)}{h\mu_k}$.

Note that the individually optimal policy is threshold type, because a server will be used if there are enough jobs waiting and it will otherwise never be used. Let $\hat{t}_j$ denote the threshold value for using server $\Pi^*(j)$. We can compute $\Pi^*(j)$, $\hat{t}_j$ and S_i^* recursively as described below.

Job 1 will use server $\Pi^*(1)$ where $\Pi^*(1) = \arg\min_{k\in S} \frac{h+\beta_k}{\mu_k}$ and $S_1^* = \Pi^*(1)$, and $\hat{t}_1 = 1$. Given $\Pi_{j-1}^* = \{\Pi^*(1), ..., \Pi^*(j-1)\}$, and given $\hat{t}_1, ..., \hat{t}_{j-1} =: l \ge j-1$, we have, for $i \le l$, $S_i^* = \Pi_k^*$ where k is such that $\hat{t}_k \le i < \hat{t}_{k+1}$ and $k \le j-1$. Consider job m, $m > l$, and suppose we have already (recursively) determined that $S_i^* = \Pi_{j-1}^*$ for $i = l, \ldots, m-1$, i.e., jobs $l, \ldots, m-1$ prefer to wait for one of the servers in Π_{j-1}^*. (We already know $S_l^* = \Pi_{j-1}^*$.) Job m will use server $k \in S \setminus \Pi_{j-1}^*$ if

$$v_m(\Pi_{j-1}^* \cup \{k\}) = \min_{u\in S\setminus\Pi_{j-1}^*} v_m(\Pi_{j-1}^* \cup \{u\}) \tag{10}$$

$$\text{and } v_m(\Pi_{j-1}^* \cup \{k\}) \le v_m(\Pi_{j-1}^*) \Leftrightarrow m \ge C^k(\Pi_{j-1}^*). \tag{11}$$

In this case $S_m^* = \Pi_{j-1}^* \cup \{k\}$, $\Pi^*(j) = k$, and $\hat{t}_k = m$. Otherwise $S_m^* = \Pi_{j-1}^*$ and we repeat with job $m+1$.

The computation above is from the perspective of the individual jobs. Alternatively, we can compute Π_j^* and $\hat{t}_j$ given Π_{j-1}^* and $\hat{t}_1, ..., \hat{t}_{j-1}$ more compactly as follows. Note that to have k and m such that $k = \Pi^*(j)$ and $m = \hat{t}_j$, then m is the smallest value greater than l such that both equations (10) and (11) hold. From (10) we can compute $\Pi^*(j)$ from $\hat{t}_j$ for $j \ge 2$:

$$\Pi^*(j) = \arg\min_{k\in S\setminus\Pi_{j-1}^*} v_{\hat{t}_j}(\Pi_{j-1}^* \cup \{k\}) = \arg\min_{k\in S\setminus\Pi_{j-1}^*} \frac{\hat{t}_j h + B(\Pi_{j-1}^*) + \beta_k}{M(\Pi_{j-1}^*) + \mu_k}, \tag{12}$$

using Lemma 11. Also, $\hat{t}_j = \min_{k\in S\setminus\Pi_{j-1}^*} \hat{t}_k(\Pi_{j-1}^*)$ where $\hat{t}_k(\Pi_{j-1}^*)$ is the optimal threshold for server $k \in S \setminus \Pi_{j-1}^*$ if only the servers in $\Pi_{j-1}^* \cup \{k\}$ exist (the others are removed from the problem). From (11),

$$\hat{t}_k(\Pi_{j-1}^*) = \min\{m : v_m(\Pi_{j-1}^* \cup \{k\}) \le v_m(\Pi_{j-1}^*), m > l\} = \max\left\{j, \left\lceil C^k(\Pi_{j-1}^*) \right\rceil\right\}, \tag{13}$$

where the second equality follows from Lemma 11. Therefore,

$$\hat{t}_j = \min_{k\in S\setminus\Pi_{j-1}^*} \max\left\{j, \left\lceil C^k(\Pi_{j-1}^*) \right\rceil\right\}. \tag{14}$$

Summarizing, we have the following theorem.

Theorem 12: With reassignment and no arrivals the individually optimal policy is determined recursively from (14) and (12), with $\Pi^*(1) = \arg\min_{k \in S} \frac{h+\beta_k}{\mu_k}$ and $\hat{t}_1 = 1$.

Although determination of the optimal policy is surprisingly complicated, it is easily implemented in a decentralized fashion by the jobs according to their priority order, and it depends on the individual parameters of the previously selected servers only through their sums, and does not depend on the thresholds for these servers.

B. The Socially Optimal Policy

We are primarily interested in minimizing the total expected cost summed over all jobs. Kumar and Walrand [33] showed that if the socially optimal policy for a scheduling problem can be implemented by each job implementing some individual policy, and if under that individual policy a job never utilizes a previously declined server, then that individual policy is, in fact, individually optimal, even if there are arrivals. We show, in Theorem 13 and later results, the converse, that the individually optimal policy is also socially optimal.

For the socially optimal policy, since jobs are indistinguishable, the only state information we need at any given time is the set of idle servers and the number of waiting jobs, and the decision is whether or not to assign a job to an available server based on the state. From a social perspective, it doesn't matter which job is assigned. It will be convenient to refer to "social cost," the total expected cost for all jobs, and "social decisions," the decision about assigning a job to a server without regard to which job. Thus, for any policy that makes the same social decisions at each decision epoch, the social cost must be the same.

1) Problem with reassignment: Decision epochs occur at potential service completions, i.e., according to a Poisson process at rate $\sum_{j \in S_n} \mu_j$. If a server is busy then a potential service completion at that server will be an actual service completion; otherwise the state will not change.

Theorem 13: For the problem with reassignment, an individually optimal policy is also socially optimal.

Proof: We suppose there are a finite number of decision epochs, N, and use induction on N. Suppose the individually optimal policy is socially optimal when the number of remaining decisions is $N-1$ or fewer. (The case of $N = 1$ is easy.) Let π^* denote the individually optimal policy for the N-stage problem (or, more accurately, it is the policy that makes the same social decisions as that of the individually optimal policy with arbitrary priorities given to jobs). Let f be another policy which makes a different social decision from the social decision of the individually optimal policy at the initial decision epoch (time 0) in the N-stage problem. Suppose the times of decision epochs and which server (potentially) completes at each epoch are coupled for the two policies. By the induction hypothesis, we may assume that f follows the individually optimal policy at the next decision epoch (time 1), and thereafter. That is, starting at time 1, f can be implemented by giving the jobs arbitrary priorities and allowing them to make their individually optimal decisions from time 1 on. Also, we can think of π^* as assigning arbitrary jobs to servers in agreement with the individually optimal policy at time 0, and then at time 1 giving the jobs arbitrary priorities and allowing them to make their individually optimal decisions from time 1 on. If f and π^* assign jobs to the same set of servers at time 0, then $f = \pi^*$ and we are done. Otherwise f and π^* may differ for several servers. We construct a policy f' below that is closer to π^* than f by moving one job (either from one server to another, or from a server to the queue, or from the queue to a server) and that has lower social cost. By repeating the construction we obtain π^* and we will be done.

First suppose that policy f assigns a job to server k and not to server j, while policy π^* assigns a job to server j and not to server k, so j is preferred to k under the individually optimal policy. Suppose, without loss of generality, that f gives the job it assigns to k the lowest priority from time 1 on (call it job n). Consider a policy f' which is identical to f except that it assigns a job to j instead of k, and it gives that job lowest priority starting from time 1 (also call it job n). Let f' assign the same priorities to the other jobs at time 1 as f does. Then all the other jobs will have the same cost under f and f' because job n has no effect on those jobs (because, for those jobs, both f and f' make the same decisions at time 0, and they both follow the individually optimal policy thereafter). Also, job n is better off under f' than under f. This is because, by definition, given the individually optimal policy will be followed from time 1 on, at time 0 it is individually optimal for some job to choose server j, but server k will be rejected by all jobs at time 0, and therefore, under the individually optimal policy, the lowest priority job would choose to use server j at time 0 if offered it, and it would reject server k.

Now suppose there are no two servers k and j such that f assigns a job to server k and not to server j, while policy π^* assigns a job to server j and not to server k. Then, if $f \neq \pi^*$, we must have $l_f \neq l_{\pi^*}$ where l_f and l_{π^*} are the number of jobs assigned to servers under f and π^* respectively at time 0.

If $l_f > l_{\pi^*}$, consider server k that is busy under f and idle under π^*. Suppose, without loss of generality, that f gives the job it assigns to k the lowest priority from time 1 on (call it job n). Consider a policy f' which is identical to f except that it does not assign a job to server k at time 0, and it gives a job in the queue the lowest priority starting at time 1 (also call it job n). After the first decision epoch both policies agree with the individually optimal policy, with priorities assigned to the other jobs arbitrarily.

Under the individually optimal policy π^* server k is idle, i.e., the lowest priority job would prefer to wait for one of the more preferred servers than to use server k, i.e., job n is better off under f'. Again, all the other jobs will have the same cost under f and f' because job n has no effect on them, and f' has smaller social cost than f.

If $l_f < l_{\pi^*}$, the construction of f' with smaller social cost than f is similar.

If $f' = \pi^*$ then we are done. Otherwise, we can repeat the argument starting with f', and comparing its social decisions at time 0 with those of π^*, where we can again think of arbitrarily assigning priorities to jobs starting at time 1. ■

2) Problem without reassignment: We would like to use basically the same proof as in Theorem 13 to show that the theorem also holds without reassignment. A key property of that proof is that job n has no effect on other jobs under the individually optimal policy, but this will not be the case in the problem without reassignment as presented earlier, where we assumed no preemption. We therefore introduce a *preemptive* individually optimal policy for the problem without reassignment. It will turn out that in the clearing system the two policies will be equivalent (because higher priority jobs will never want to preempt lower priority jobs), but they will differ when we have arrivals, and we will need to allow preemption in order for the individually optimal policy to be socially optimal. The social decisions of the individually optimal policy with preemption can still be implemented without preemption, because the service times of all jobs are identically and exponentially distributed, so replacing one job with another on a server has no social impact.

We now assume that a higher priority job that is not currently on a server (it has just been preempted, or it is otherwise in the queue) can preempt a lower priority job on a server. In contrast to the problem with reassignment, here a job cannot leave a server unless preempted (or it completes service). We will show that the preemptive and nonpreemptive individually optimal policies are identical in terms of the social decisions and social costs, and that they are both socially optimal.

Lemma 14: For any given initial social state, a preemptive individually optimal policy makes the same social decisions (and has the same social costs) regardless of the priorities of jobs initially assigned to servers.

Proof: Note that the problem faced by individual jobs is the same with and without preemption, except that with preemption the job can assume servers with lower priority jobs are available. Hence, under a preemptive individually optimal policy, if there are at least $t_1 = 1$ jobs waiting, the highest priority waiting job will use server 1 if it is idle or if it is serving a lower priority job. This is repeated for each server in order, so that, after servers $1, ..., j$ are considered, the t_jth priority job in queue uses server j if it is idle or serving a lower priority job. Therefore two preemptive individually optimal policies will be socially identical after the first decision regardless of the priorities of jobs initially assigned to servers. ■

Corollary 15: The individually optimal preemptive policy can be implemented without preemption for the problem without reassignment.

Proof: Without loss of generality by Lemma 14, let the servers that are initially busy have the highest priority jobs under the preemptive individually optimal policy. Then those jobs will not be preempted, and later jobs will also not be preempted because of Lemma 1. ■

Corollary 16: : When preemption is not permitted, the individually optimal policy makes the same social decisions as the preemptive individually optimal policy.

Proof: Recall that for the original problem without reassignment (and without preemption), we only gave priority labels to the n_0 jobs initially in the queue, and not to the k, say, jobs on the k busy servers. We can equivalently label the k jobs on servers as the k highest priority jobs, and give the n_0 jobs in the queue priorities $k+1, ..., k+n_0$, consistent with the labeling when preemption is permitted. Now suppose we permit jobs in queue to preempt lower priority jobs on servers. From the argument of Corollary 15, the individually optimal preemptive policy will not preempt, so it is still optimal even when preemption is not permitted. ■

Finally we have the following theorem, where the first part follows using the same proof as in Theorem 13, and the second part follows from Corollary 16.

Theorem 17: For the problem without reassignment, a preemptive individually optimal policy is also socially optimal. Hence, for the original nonpreemptive model, the individually optimal policy is also socially optimal.

C. Important Special Cases

1) Agreeable speeds and costs: For the problem with reassignment, the optimal policy is simpler if $\beta_j \leq \beta_{j+1}$ and $\mu_j \geq \mu_{j+1}$, $j = 1, ..., s-1$, so faster servers are also cheaper, and have smaller index, i.e., $\Pi^*(j) = j, j = 1, ..., s$. (Indeed, as we show below, all we need is $\mu_j \geq \mu_{j+1}$ and $\frac{h+\beta_j}{\mu_j} \leq \frac{h+\beta_{j+1}}{\mu_{j+1}}$). In this case, with reassignment,

$$\hat{t}_j = \max\{j, \lceil C_j \rceil\}, \tag{15}$$

where $C_j = C^j(1, ..., j-1)$ for ease of notation.

Now let us compare the thresholds for $j \geq 2$ with and without reassignment. From (4), the threshold without reassignment is

$$t_j = \max\left\{t_{j-1}, \left\lceil C_j + \frac{M_{j-1}}{\mu_j} - (j-1)\right\rceil\right\}. \tag{16}$$

Recall that without reassignment, we only considered those jobs in queue in our priority labeling, whereas with reassignment, *all* jobs present have priority labeling. Let $a_j = C_j + \frac{M_{j-1}}{\mu_j} - (j-1)$.

Lemma 18: If $\beta_j \leq \beta_{j+1}$ and $\mu_j \geq \mu_{j+1}$, $j = 1, ..., s-1$, then a_j is increasing in j.

Proof:

$$\begin{aligned} a_j &= \frac{\beta_j M_{j-1} - \mu_j B_{j-1}}{h\mu_j} + \frac{M_{j-1}}{\mu_j} - j + 1 \\ &= \frac{\beta_j M_{j-2} - \mu_j B_{j-2}}{h\mu_j} + \frac{\beta_j \mu_{j-1} - \mu_j \beta_{j-1}}{h\mu_j} + \frac{M_{j-2}}{\mu_j} + \frac{\mu_{j-1}}{\mu_j} - j + 1 \\ &\geq \frac{\beta_{j-1} M_{j-2} - \mu_{j-1} B_{j-2}}{h\mu_{j-1}} + 0 + \frac{M_{j-2}}{\mu_{j-1}} + 1 - j + 1 = a_{j-1} \end{aligned} \tag{17}$$

■

Therefore, with $\beta_j \leq \beta_{j+1}$ and $\mu_j \geq \mu_{j+1}$, the threshold without reassignment is

$$t_j = \left\lceil C_j + \frac{M_{j-1}}{\mu_j}\right\rceil - (j-1), \tag{18}$$

and the threshold for the slow-server problem without reassignment ($\beta_j \equiv 0$) is

$$t_j^0 = \left\lceil \frac{M_{j-1}}{\mu_j}\right\rceil - (j-1). \tag{19}$$

Suppose that with reassignment the threshold for using the jth server is such that the server will not automatically be used if there are enough jobs, that is, $\hat{t}_j = \lceil C_j \rceil$. Then we have the interesting fact that the threshold value without reassignment, t_j, is the sum of the threshold value with reassignment, $\hat{t}_j$, plus the threshold value for the slow-server problem without reassignment, t_j^0 (within one, due to the ceiling function). For instance, suppose $s = 2$ and $(h, \beta_1, \beta_2, \mu_1, \mu_2) = (1, 6, 3, 3, 1)$. Then $t_2^0 = 3$, $\hat{t}_2 = \lceil C_j \rceil = 3$ and $t_2 = 6 = t_2^0 + \hat{t}_2$.

2) Energy as a function of speed: In practical systems, $P(\mu) = c_1 + c_2\mu^\alpha$ where $P(\mu)$ is the power (energy consumed per unit time) required to run the servers at speed μ i.e., $\beta_j = c_1 + c_2\mu_j^\alpha$. In many applications $\alpha \in (1, 3)$, for instance for CMOS chips, $\alpha \approx 1.8$ is a good approximation [27], and $\alpha = 1$ or 2 is often assumed in the literature.

If $\alpha = 1$, i.e., $\beta_j = c_1 + c_2\mu_j$, then the $\frac{h+\beta_j}{\mu_j}$ index will be strictly decreasing in μ_j. In this case, faster servers will be preferred. For the models without and with reassignment respectively, regardless of c_2,

$$t_j = \max\left\{t_{j-1}, \left\lceil \frac{c_1 + h}{h}\left(\frac{M_{j-1}}{\mu_j} - (j-1)\right)\right\rceil\right\}, \tag{20}$$

$$\hat{t}_j = \max\left\{j, \left\lceil \frac{c_1}{h}\left(\frac{M_{j-1}}{\mu_j} - (j-1)\right)\right\rceil\right\}. \tag{21}$$

Specifically, for $c_1 = 0$, i.e., proportional usage costs and speeds, we have $t_j = t_j^0$, so the thresholds are the same as the thresholds for the slow-server problem.

If $\alpha = 2$, i.e., $\beta_j = c_1 + c_2\mu_j^2$, then the $\frac{h+\beta}{\mu}$ index will have its global minimum at $\mu = \sqrt{\frac{h+c_1}{c_2}}$. Therefore if $\mu_j \leq \sqrt{\frac{h+c_1}{c_2}}$ for all j, then ordering servers in increasing order of the $\frac{h+\beta_j}{\mu_j}$ index will be equivalent to ordering faster servers first. Let $M_j^{(2)} = \sum_{i=1}^{j} \mu_i^2$. Then,

$$t_j = \max\left\{t_{j-1}, \left\lceil \frac{c_1 + h}{h}\left(\frac{M_{j-1}}{\mu_j} - (j-1)\right) - \frac{c_2}{h}\left(M_{j-1}^{(2)} - \mu_j M_{j-1}\right)\right\rceil\right\} \tag{22}$$

$$\hat{t}_j = \max\left\{j, \left\lceil \frac{c_1}{h}\left(\frac{M_{j-1}}{\mu_j} - (j-1)\right) - \frac{c_2}{h}\left(M_{j-1}^{(2)} - \mu_j M_{j-1}\right)\right\rceil\right\}. \tag{23}$$

Note, for fixed c_1, the thresholds are smaller when operating costs are quadratic in the speeds than when they are linear, regardless of c_2.

III. The System with Arrivals

We suppose that new jobs arrive according to a Poisson process with arrival rate $\lambda < \sum_{i=1}^{s} \mu_i$, and our objective is to minimize the long-run average cost per job. Now, for the individually optimal policy, all jobs, even those being served, have priority labels, and any jobs present at time 0, including those on servers, are given arbitrary job priorities. Arriving jobs are given highest priority, and the other jobs are relabeled so that jobs that arrived later have priority over those that arrived earlier (or were initially present). When reassignment is permitted, *any* higher priority job may preempt any lower priority job on a server (regardless of whether it is already on a server); when reassignment is not permitted, a higher priority job *that is not currently on a server* (because it has just arrived, it has just been preempted, or it is otherwise in the queue) can preempt a lower priority job on a server. As we noted in the last section, from the social perspective, the problem without reassignment is socially equivalent to allowing high priority jobs in queue to preempt lower priority jobs on servers because we have identical jobs and exponential service times. Our priority discipline for both models is the last-come-first-priority discipline, with preemption (LCFP-P) of Xu [14].

We first show that when priorities are given according to LCFP-P, with reassignment the individually and socially optimal policies coincide and are threshold policies. Without reassignment the individually optimal policy is still a threshold policy, but the individually and socially optimal policies coincide only for two servers.

A. The Socially Optimal Policy

1) Problem with reassignment:

Theorem 19: Under LCFP-P with reassignment, the individually optimal policy is threshold type and the most preferred server is always used by the highest priority job.

Proof: At every decision epoch (which now occur according to a Poisson process at rate $\lambda + \sum \mu_j$), all servers are offered to the jobs in priority order. Each job will have a preference order for the servers and it will either choose to use the preferred server among the available ones or wait for one of the busy servers. Let v_i be the individually optimal cost of the ith priority job. It is clear that idling is not optimal for job 1, because idling and using a server later is costlier than using the server now, so it will use a server, and that server will by definition be the most preferred server (the one used if there is at least one job), call it server 1. We will show that v_i is strictly increasing in i. Therefore, eventually (if there are enough jobs present) there will be a job choosing to use some server besides server 1, call it server 2, and so on for the remaining servers. If job i chooses to use a server, then that server will be used any time there are at least i jobs, i.e., the individually optimal policy will be threshold type.

We now argue that the thresholds will all be finite. If not, there will be a set of servers that will never be used, say set $S_\emptyset$. Let t be the threshold such that all servers in $S \backslash S_\emptyset$ will be used. Then, for $i > t$, $v_i = \frac{h}{\sum_{k \notin S_\emptyset} \mu_k - \lambda} + v_{i-1} > v_{i-1}$, where $\frac{1}{\sum_{k \notin S_\emptyset} \mu_k - \lambda}$ is the length of a busy cycle in an $M/M/1$ queue with service rate $\sum_{k \notin S_\emptyset} \mu_k$ [34]. (Note that $S_\emptyset$ is such that $\sum_{k \notin S_\emptyset} \mu_k > \lambda$, because the average cost will be infinite otherwise). This is because, until job i becomes job $i-1$, all servers will be busy (and acting as a single fast server because of the exponential service times) and all arrivals while job i is waiting to become job $i-1$ must be served. Therefore, v_i is strictly increasing in i, so, for i large enough, the cost for using the kth server to completion, for some $k \in S_\emptyset$, is $\frac{h+\beta_k}{\mu_k} < v_i$. Thus, the threshold for the kth server is finite and the kth server will be used if the queue length is sufficiently large. On the other hand, we also have that v_i is finite, because it could accept any server that becomes available, in which case its cost would be bounded above by the holding cost for waiting for i busy periods assuming all servers are used (none is offered to the considered job) plus $\max_j \beta_j/\mu_j$. ■

Next we consider the socially optimal policy.

Theorem 20: The individually optimal policy is also socially optimal.

Proof: Note that under the LCFP-P priority rule, we still have that the lowest priority job will have no impact on the other jobs. Hence, the proof is similar to the proof of Theorem 13 with π^* and f similarly defined. The only difference is that there are future arrivals after the initial decision epoch. These arrivals will have the same individual expected cost in π^* and f, because they have higher preemptive priority, and both policies follow the individually optimal policy after the first decision. Therefore, arguing as in the proof of Theorem 13, π^* will still be better than f for all jobs, and hence it will be better in terms of the total expected cost. ■

2) Problem without reassignment: The same type of argument as the proof of Theorem 19 also gives us the following, where here the threshold for using the most preferred available servers will generally depend on the states of less preferred servers (idle or, if busy, the priority of the job on the server).

Theorem 21: Under LCFP-P without reassignment, the individually optimal policy is threshold type and the most preferred server is always used by the highest priority job.

a) Two servers:

Theorem 22: The individually optimal policy is also socially optimal when there are only two servers.

Proof Outline: For the details, we refer to our proofs of Theorems 13 and 20 and Xu's [14] proofs for the slow-server problem with two servers and arrivals. The idea is as follows. For the infinite horizon problem, it is both socially and individually optimal for the preferred server (call it server 1) to always be used, because waiting and then using it just incurs extra costs, and, for the individually optimal policy, the highest priority job will use server 1. Therefore, suppose server 2 is available and the lowest priority job, job $n > 1$, is in the queue. We can argue as we did earlier that deviating from the individually optimal policy for job n for the first decision, and then following the individually optimal policy, hurts job n, and has no effect on the other jobs. ■

b) More than two servers: Theorem 22 does not hold for $s > 2$ (see Theorem 24 below). In particular, for arrivals with $s > 2$, we no longer have Lemma 14, that for any initial state a preemptive individually optimal policy makes the same social decisions regardless of the priorities of the jobs initially assigned to servers. We also no longer have that the decision of the lowest priority job in *queue* (call it job n) has no impact on the other jobs. Consider a three-server case where at the initial decision epoch server 1 (the most preferred server) is busy and f assigns job n to server 2 (the second most preferred server), while server 2 is idle under π^*. Then we cannot say that all the jobs except job n have the same individual expected costs in both policies, because if a job with lower priority than job n is using server 3, it will be affected by job n's decision. For instance under π^*, job n can become job $n+1$ due to an arrival, and it might choose to preempt the job using server 3. However, under f, job n will be on a server, or may have completed service, so cannot preempt another job, and then the job on server 3 could be better off under f.

We show, with a finite horizon example similar to Weber's example for the slow-server problem, that for $s > 2$, if a threshold policy is optimal, the threshold for using a server depends on the states of less desirable servers [21]. In our example the only source of heterogeneity is in usage costs.

Example 23: Suppose $s = 3$, $\mu_j \equiv \mu$ and $\beta_1 \leq \beta_2 \leq \beta_3$. Without loss of generality we suppose $\lambda + 3\mu = 1$. Let $V_N(k, \alpha)$ denote the total cost accrued when there are N decision epochs to go starting from the state (k, α), where k denotes the number waiting in queue and $\alpha \in \{0, 1\}^3$ denotes the state of the servers (0 for idle and 1 for busy). Note that the decision epochs are arrivals or service completions. Let $A_j\alpha$ denote the vector obtained from α by assigning jobs to the j cheapest available servers and let $D_j\alpha$ denote the vector obtained from α by setting α_j to 0. We can write the dynamic programming equations as

$$V_{N+1}(k, \alpha) = \min_{j \leq k \wedge (3 - |\alpha|)} \bar{V}_{N+1}(k - j, A_j\alpha) \tag{24}$$

where $|\cdot|$ denotes the L^1 norm, $\wedge$ denotes minimum, and

$$\bar{V}_{N+1}(k, \alpha) = h(k + |\alpha|) + \beta^T\alpha + \lambda V_N(k + 1, \alpha) + \mu \sum_{j=1}^{3} V_N(k, D_j\alpha). \tag{25}$$

Suppose $(\lambda, \mu, h, \beta_1, \beta_2, \beta_3) = (0.25, 0.25, 3.01, 1.50, 7.50,\ 9.50)$. Then for $N = 24$ when server 2 is the cheapest available server and one job is waiting, we get the following values, with four decimal places of accuracy:

$$V_{24}(1, 100) = \min\{\bar{V}_{24}(1, 100), \bar{V}_{24}(0, 110)\} = \min\{191.9873, 191.9899\} = \bar{V}_{24}(1, 100) \tag{26}$$

$$V_{24}(1, 101) = \min\{\bar{V}_{24}(1, 101), \bar{V}_{24}(0, 111)\} = \min\{242.0680, 242.0582\} = \bar{V}_{24}(0, 111) \tag{27}$$

Thus, when the most expensive server 3 is idle and there is only one job waiting, we would prefer not to assign that waiting job to server 2, whereas when server 3 is busy we would prefer to assign the waiting job to server 2. We found, as did Weber, that it is hard to come up with an example where the optimal threshold depends on the server states. For example the threshold for the second server does not depend on the state of the third server when the parameter values are slightly different: $(\lambda, \mu, h, \beta_1, \beta_2, \beta_3) = (0.25, 0.25, 3.00, 1.50, 7.50, 9.50)$, or when the horizon length is slightly different.

Because the socially optimal policy for server 2 depends on the state of server 3 in this example, the socially optimal policy cannot be the same as the individually optimal policy. To see this, suppose they coincide. Then, when there are two jobs with job 1 on server 1 and job 2 in the queue (servers

2 and 3 available), it would have to be individually optimal for job 2 to wait in queue. Because job 2's individually optimal policy does not depend on lower priority jobs, job 2 would make the same individually optimal decision if a third, lower priority job, were using server 3. However, if job 1 were using server 1, job 2 were using server 3, and job 3 were in the queue, job 3 would use server 2 if its individually optimal policy were the same as the socially optimal policy. Thus, given the same social state $(1, 101)$, the individually optimal decision of the job in queue depends on its priority relative to the job on server 3, i.e., its social decision is different. Thus we have Theorem 24 below.

Theorem 24: For the problem without reassignment and more than two servers, the individually optimal policy under LCFP-P is not necessarily the same as the socially optimal policy.

B. The Optimal Policy with Two Servers

Now we more completely characterize the optimal policy for two servers and arrivals, and we show how the policy depends on the arrival rate. For the problem with reassignment we give an explicit algorithm for computing the optimal policy. We have already shown that the individually optimal policy is socially optimal; it will be convenient to consider the optimal policy from the individual perspective in this section. Assume throughout this section that $\frac{h+\beta_1}{\mu_1} \leq \frac{h+\beta_2}{\mu_2}$.

1) Problem with reassignment: In Section II we showed that server 1 is preferred to server 2 for all states. However, with arrivals the preference depends on λ. For example, for $\mu_1 = 0.25$, $\mu_2 = 0.75$, $\beta_1 = 2$, $\beta_2 = 26$, $h = 8$, if $\lambda = 0$ the first server is preferred, but if $\lambda = 2$, the second server is preferred. To illustrate this example further we consider $\lambda = 2$, but with a model intermediate between the clearing system and the system with infinite Poisson arrivals, i.e., a model with a finite number of arrivals, and exponential interarrival times. It will also be convenient to use this model with induction on the number of arrivals in some of our later proofs. Let v_i^m denote the individually optimal cost for the job currently in the ith priority order when there are m future arrivals and assuming both servers are available. For our example, we suppose there will be at most two arrivals; after that arrivals stop. This example shows the somewhat surprising result that it can be optimal to use server 1 but not server 2 in some states, and to use server 2 but not server 1 in others.

Example 25: Suppose there are at most two future arrivals and the system parameters are $(\mu_1, \mu_2, \beta_1, \beta_2, h, \lambda)=$ $(0.25, 0.75, 2, 26, 8, 2)$. We know from Theorem 12 that, without arrivals, v_i^0 can be calculated as

$$v_1^0 = \min\{\frac{h+\beta_1}{\mu_1}, \frac{h+\beta_2}{\mu_2}\} = \min\{40, 45.33\} = 40 \tag{28}$$

$$v_2^0 = \min\{\frac{2h+\beta_1}{\mu_1}, \frac{2h+\beta_1+\beta_2}{\mu_1+\mu_2}\} = \min\{72, 44\} = 44 \tag{29}$$

$$v_3^0 = \frac{3h + \beta_1 + \beta_2}{\mu_1 + \mu_2} = 52. \tag{30}$$

Thus, with no future arrivals ($m = 0$) server 1 will be preferred and server 2 will be used if there are at least two jobs. When $m = 1$ and at least one job is currently present, $v_1^1 = \min\{\frac{h+\beta_1+\lambda v_2^0}{\mu_1+\lambda}, \frac{h+\beta_2+\lambda v_2^0}{\mu_2+\lambda}\} = \min\{43.56, 44.36\} = 43.56$, so server 1 will be preferred by job 1, and $v_2^1 = \min\{\frac{h+\mu_1 v_1^1+\lambda v_3^0}{\mu_1+\lambda}, \frac{h+\beta_2+\mu_1 v_1^1+\lambda v_3^0}{\mu_1+\mu_2+\lambda}\} = \min\{54.62, 49.63\} = 49.63$, so server 2 will be used if there are at least two jobs currently present and one future arrival. Finally, $v_1^2 = \min\{\frac{h+\beta_1+\lambda v_2^1}{\mu_1+\lambda}, \frac{h+\beta_2+\lambda v_2^1}{\mu_2+\lambda}\} = \min\{48.56, 48.46\} = 48.46$. Thus, with a single job in the system at time 0 and two future arrivals, the job will use the second (faster but more expensive) server, but with a single job at time 0 and no new arrivals, the job will use the first server. We have also found computationally that when the arrival process does not stop, server 2 is still preferred to server 1.

The following theorem gives a sufficient condition for server 1 to be preferred to server 2 for all states and all arrival rates. An intuitively obvious condition is when server 1 is both cheaper and faster, but, in fact, we need a less restrictive condition. Note that by Theorem 22, Theorem 26 holds for both the individually and the socially optimal policies. Because the proofs of this result and the others in this section are quite technical, they are given in the appendix.

Theorem 26: Under the optimal policy with reassignment for two servers, if $\frac{h+\beta_1}{\mu_1} \leq \frac{h+\beta_2}{\mu_2}$ *and* $\mu_1 \geq \mu_2$, then server 1 is the preferred server for all states and arrival rates, and both for a finite number of arrivals and for the standard (unstopped) Poisson arrival process.

We now consider the case where $\frac{h+\beta_1}{\mu_1} \leq \frac{h+\beta_2}{\mu_2}$ but we may have $\mu_1 < \mu_2$, so, as we have seen, the server preference may depend on λ, where $\lambda < \mu_1 + \mu_2$. Let $v_i(\lambda) = \lim_{m\to\infty} v_i^m(\lambda)$ be the individually optimal expected cost when the arrivals are not stopped (regular Poisson arrivals with rate λ). Here, we make the dependence of v_i on the arrival rate explicit. We have the following results. Lemma 27 agrees with intuition due to our LCFP-P priority; a higher arrival rate means a job is more likely to move back in priority.

Lemma 27: $v_i(\lambda)$ is increasing in λ.

Theorem 28: For reassignment and two servers, either server 1 is preferred for all λ or there exists a $0 < \lambda_0 < \mu_1 + \mu_2$ such that for all $\lambda < \lambda_0$, server 1 is preferred, and for $\lambda > \lambda_0$, server 2 is preferred.

Let server A (B) be the preferred (nonpreferred) server, given arrival rate λ. That is, if $\mu_1 \geq \mu_2$ or if $\lambda < \lambda_0$, $A = 1$ and $B = 2$, else $A = 2$ and $B = 1$. Suppose the ith priority job, $i \geq 2$ is offered server B. Then it will use server B only if

$$\frac{h + \mu_A v_{i-1}(\lambda) + \mu_B v_i(\lambda) + \lambda v_{i+1}(\lambda)}{\mu_A + \mu_B + \lambda} \geq \frac{h + \beta_B + \mu_A v_{i-1}(\lambda) + \lambda v_{i+1}(\lambda)}{\mu_A + \mu_B + \lambda} \tag{31}$$

which is equivalent to $v_i(\lambda) \geq \frac{\beta_B}{\mu_B}$. Let

$$T(\lambda) = \min\{i : v_i(\lambda) \geq \frac{\beta_B}{\mu_B}\} \tag{32}$$

denote the threshold for using the nonpreferred server when the arrival rate is λ. Note that $T(\lambda)$ is finite, because otherwise server B would never be used, and we would have $v_i(0) = (ih + \beta_A)/\mu_A$, and $v_i(\lambda) \geq v_i(0)$ (from Lemma 27).

Corollary 29: $T(\lambda)$ is decreasing in λ.

Note that, from Theorem 28 and Corollary 29, if $\mu_1 < \mu_2$ and $\lambda > \lambda_0$, then $T(\lambda) = 2$, because at λ_0 we are indifferent between the two servers, so both would be used if there are at least two jobs.

We now give an algorithm to compute the optimal threshold and the optimal expected cost for job i.

Lemma 30: Suppose a threshold policy with threshold T is used, so job 1 will be assigned to server A and job T will be assigned to server B if there are at least T jobs. Then

$$v_i = \begin{cases} \dfrac{h + \beta_A + \lambda(H_2 + P_2 v_T)}{\mu_A + \lambda P_2} & \text{if } i = 1 \\ H_i + P_i v_T + (1 - P_i) v_1 & \text{if } 1 < i < T \\ \dfrac{(h + \beta_B + \mu_A H_{T-1} + G)}{\dfrac{\mu_A^2(1 - P_{T-1})}{\mu_A + \lambda P_2} + \mu_B} & \text{if } i = T \\ \dfrac{(i - T)h}{\mu_A + \mu_B - \lambda} + v_T & \text{if } i > T \end{cases} \tag{33}$$

where

$$P_i = \begin{cases} \frac{\left(\frac{\mu_A}{\lambda}\right)^{i-1} - 1}{\left(\frac{\mu_A}{\lambda}\right)^{T-1} - 1} & \text{if } \mu_A \neq \lambda \\ \frac{i-1}{T-1} & \text{if } \mu_A = \lambda, \end{cases} \tag{34}$$

$$H_i = \begin{cases} h\left[\frac{i-1}{\mu_A - \lambda} - \frac{T-1}{\mu_A - \lambda} P_i\right] & \text{if } \mu_A \neq \lambda \\ (i-1)(T-i) & \text{if } \mu_A = \lambda, \end{cases} \tag{35}$$

and

$$G = \frac{\lambda h}{\mu_A + \mu_B - \lambda} + \frac{\mu_A(h + \beta_A + \lambda H_2)(1 - P_{T-1})}{\mu_A + \lambda P_2}. \tag{36}$$

We can compute the optimal thresholds by successively computing v_T using Lemma 30 for $T = 2, 3, ...$ The first time v_T exceeds β_B/μ_B will give us the optimal threshold T.

Note that the algorithm eventually terminates by finiteness of the threshold, and the optimal threshold can also be calculated more efficiently by a binary search between 2 and $T(0)$ due to Corollary 29. In Figure 1 we plot the threshold values and λ_0 when $(h, \beta_1, \beta_2, \mu_1, \mu_2)$= $(2, 2, 26, 0.25, 0.75)$.

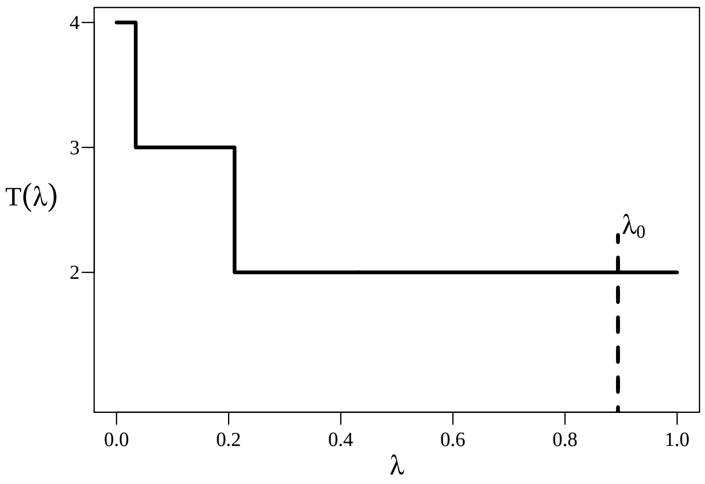

Fig. 1. Optimal threshold value vs arrival rate.

2) Problem without reassignment: As with reassignment, the preference might depend on the arrival rate. In this case a job's individually optimal expected cost will not only depend on its priority order but also will depend on whether it is in service or not, and the states of the other servers. The conclusion of Example 25 (with the same numbers) will still be true, so that the second server will be preferred to the first server in some states but the reverse will be true in others. The following sufficient condition for server 1 to always be preferred is the same as for the problem with reassignment, and the proof is similar.

Theorem 31: For the two server case without reassignment, if $\frac{h+\beta_1}{\mu_1} \leq \frac{h+\beta_2}{\mu_2}$ and if $\mu_1 \geq \mu_2$, then server 1 is preferred to server 2 for all states and for all λ.

We would like to show that Theorem 28 and Corollary 29 also hold without reassignment, but we were only able to show the partial versions below. Suppose the highest priority job prefers to use server A when the arrival rate is λ. Call the other server B.

Theorem 32: Without reassignment and two servers, either server 1 is preferred for all λ or there exists a $0 < \lambda_0 < \mu_A + \mu_B$ such that for all $\lambda < \lambda_0$, server 1 is preferred.

Corollary 33: Let $\mathcal{L} \subseteq (0, \infty)$ be the set of arrival rates where the highest priority job prefers server A and let $T(\lambda)$ denote the threshold for using server B when the arrival rate is $\lambda \in \mathcal{L}$. If $\mu_A \leq \mu_B$, then $T(\lambda)$ is decreasing in λ on $\mathcal{L}$.

Note that from Theorems 31 and 32, Corollary 33 is equivalent to saying that $T(\lambda)$ is decreasing for all λ if $\mu_1 = \mu_2$ (so the preferred server is $A = 1$), and it is also decreasing for λ such that server 1 is preferred and $\mu_1 < \mu_2$.

To show Theorem 32 and Corollary 33 we need some additional definitions and preliminary lemmas. Assuming uniformization with rate $\lambda + \mu_A + \mu_B = 1$ and a finite horizon on the number of decision epochs, let $w_i^N(\lambda)$ denote the individually optimal cost for N decision epochs for job i if it is in queue and if server B is either idle or serving a lower priority job than job i. Let $x_i^N(\lambda)$ be similarly defined assuming job i is using server B. Clearly $x_i^N(\lambda) \geq w_i^N(\lambda)$, and $x_i^N(\lambda) = w_i^N(\lambda)$ for $i \geq T^N(\lambda)$, where $T^N(\lambda)$ is the threshold for using server B. Because reassignment is not permitted, we must define both $x_i^N(\lambda)$ and $w_i^N(\lambda)$, instead of the single $v_i^N(\lambda)$ that we used in the problem with reassignment. This complicates the analysis, and also makes our conclusions less general. Under the individually optimal policy, job $T^N(\lambda)$ will accept server B when offered, where

$$\begin{aligned} T^N(\lambda) &= \min \left\{ \begin{array}{l} i : h + \beta_B + \lambda x_{i+1}^N(\lambda) + \mu_A x_{i-1}^N(\lambda) \\ \quad \leq h + \lambda w_{i+1}^N(\lambda) + \mu_A w_{i-1}^N(\lambda) + \mu_B w_i^N(\lambda) \end{array} \right\} \\ &= \min \left\{ \begin{array}{l} i : \beta_B \leq -\lambda[x_{i+1}^N(\lambda) - w_{i+1}^N(\lambda)] \\ \quad -\mu_A[x_{i-1}^N - w_{i-1}^N(\lambda)] + \mu_B w_i^N(\lambda) \end{array} \right\}. \end{aligned} \tag{37}$$

Theorem 32 and Corollary 33 will follow from the lemmas below. (Note that the right hand side of (37) is increasing in λ from Lemma 36 below.) The proof of the first lemma is trivial.

Lemma 34: $w_i^N(\lambda)$ and $x_i^N(\lambda)$ are both increasing in N and i.

Lemma 35: $x_i^N(\lambda) - w_i^N(\lambda)$ is decreasing in i.

Lemma 36: Let $\mathcal{L} \subseteq (0, \infty)$ be the set of arrival rates where the highest priority job prefers server A. If $\mu_A \leq \mu_B$ then
(i) $x_i^N(\lambda) - w_i^N(\lambda)$ is decreasing in λ on $\mathcal{L}$,
(ii) $w_i^N(\lambda)$ and $x_i^N(\lambda)$ are both increasing in λ on $\mathcal{L}$.

IV. Extensions

No holding cost while in service: Suppose the only cost incurred while a job is in service is the usage cost, i.e., the holding cost is incurred only when a job is waiting in queue. All the results discussed in the paper can easily be extended to cover this case. Lemmas 3 and 4 and Theorem 5 would be modified as follows. Now server j is preferred to server k if $\beta_j/\mu_j \leq \beta_k/\mu_k$, $v_i(\underline{S}) = \frac{ih+B_{\sigma_i}}{M_{\sigma_i}} =: v_i(\sigma_i)$, and $t_j = \max\{t_{j-1}, \lceil C_j \rceil\}$, $j = 2, ..., s$, with $t_1 = 1$.

Discounted costs and impatient jobs: Our results still hold if costs are exponentially discounted, or if each customer abandons the system after waiting an exponentially distributed amount of time, independently of the other jobs.

General arrival processes and batch arrivals: Suppose the arrival process is state and policy independent, but arbitrary otherwise. Also suppose the long-run arrival rate is less than the sum of the service rates. In this case, explicit calculations of thresholds and individual expected costs will not be possible. However, the results stating that the individually optimal policy under LCFP-P is a (state-dependent) threshold policy and that the socially optimal policy is the same as the individually optimal policy (except for the problem without reassignment, with arrivals and with more than two servers), will still hold for general arrival processes. Similar results hold if the arrivals occur in batches with independent and identically distributed sizes, where priorities within a batch are assigned arbitrarily but all jobs within a batch have higher preemptive priority over all jobs that arrived earlier.

V. Conclusion

We found the optimal policy for minimizing overall energy and holding costs in a problem where servers are heterogeneous in both their speeds and their energy costs, by considering an alternate problem where jobs make their own decisions to minimize their own costs subject to a priority rule that makes the externality of the lowest priority job zero. We showed that the optimal policies for social and individual optimization coincide regardless of whether jobs can be reassigned or not, except in the problem with more than two servers, Poisson arrivals, and with no reassignment. By considering individual optimization, we showed that the optimal policy is threshold type, and we were able to compute the optimal thresholds for clearing systems, and for the two-server system with arrivals and reassignment permitted. Surprisingly, with arrivals, the preference ordering of servers depends on the arrival rate, and cannot be reduced to a simple index rule. Our results extend to impatient customers, exponential discounting, and more general arrival processes.

VI. Acknowledgements

We appreciate the feedback from the referees and associate editor, which led to a much improved presentation.

APPENDIX

PROOF OF THEOREM 26

Because we consider costs under the individually optimal policy, it is easy to show that v_i^m is increasing in i and m. To show that server 1 will be preferred, we prove that, for all m,

$$\begin{aligned} v_1^m &= \min\{\frac{h+\beta_1+\lambda v_2^{m-1}}{\mu_1+\lambda}, \frac{h+\beta_2+\lambda v_2^{m-1}}{\mu_2+\lambda}\} \\ &= \frac{h+\beta_1+\lambda v_2^{m-1}}{\mu_1+\lambda}. \end{aligned} \tag{38}$$

This is equivalent to showing

$$h\mu_2+\beta_1\mu_2+\beta_1\lambda+\mu_2\lambda v_2^{m-1} \le h\mu_1+\beta_2\mu_1+\beta_2\lambda+\mu_1\lambda v_2^{m-1}. \tag{39}$$

Because $\frac{h+\beta_1}{\mu_1} \le \frac{h+\beta_2}{\mu_2}$, it is sufficient to show that

$$(\mu_1-\mu_2)v_2^{m-1} \ge \beta_1-\beta_2. \tag{40}$$

Since v_2^{m-1} is increasing in m, and $\mu_1 \ge \mu_2$, it is enough to show (40) for $m=1$.

To calculate v_2^0 first suppose $2 \ge \frac{\beta_2\mu_1-\beta_1\mu_2}{h\mu_2}$. Then by Theorem 12 the second job is above the threshold and will use the second server, so we have $v_2^0 = \frac{2h+\beta_1+\beta_2}{\mu_1+\mu_2}$. Therefore,

$$\begin{aligned} (\mu_1-\mu_2)v_2^0 &= \frac{2(h\mu_1-h\mu_2)+(\mu_1-\mu_2)(\beta_1+\beta_2)}{\mu_1+\mu_2} \\ &\ge \frac{2(\beta_1\mu_2-\beta_2\mu_1)+(\mu_1-\mu_2)(\beta_1+\beta_2)}{\mu_1+\mu_2} \\ &= \frac{\beta_1\mu_2-\beta_2\mu_1+\mu_1\beta_1-\mu_2\beta_2}{\mu_1+\mu_2} \\ &= \beta_1-\beta_2, \end{aligned} \tag{41}$$

where the inequality follows from $\frac{h+\beta_1}{\mu_1} \le \frac{h+\beta_2}{\mu_2}$. Hence, (40) is true.

The argument when $2 < \frac{\beta_2\mu_1-\beta_1\mu_2}{h\mu_2}$ is similar, and (40) follows.

PROOF OF LEMMA 27

Choose $\lambda_1 < \lambda_2 < \mu_1+\mu_2$; we will show that $v_i(\lambda_1) \le v_i(\lambda_2)$. We use uniformization with rate $\lambda_2+\mu_1+\mu_2=1$ for the decision epochs. Let $w_i^N(\lambda)$ be the individually optimal cost of the ith job for N decision epochs, and $w_i^0(\lambda)=0$. Because a job will leave the system in a finite number of decision epochs with probability 1, we have $\lim_{N\to\infty} w_i^N(\lambda) = v_i(\lambda)$ for all $i<\infty$. It is easy to show that

$$w_i^N(\lambda) \text{ is increasing in } N,\ i. \tag{42}$$

We will show that $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$ by induction on N, where the $N = 0$ case is trivial. Suppose $w_i^{N-1}(\lambda_1) \leq w_i^{N-1}(\lambda_2)$ for all i. We show $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$ by using a second induction on i. First consider $i = 1$. In both systems (with arrival rates λ_1 and λ_2), job 1 will be able to use either server, or, because the horizon is finite, it may choose to wait. Let us first suppose that job 1 chooses to wait in system 2 (because N is small and β is large), so $w_1^N(\lambda_2) = Nh$. (It is easy to show that if job 1 waits at stage N, then it will continue waiting until the end of the horizon.) Then because job 1 can also wait in system 1 we have $w_1^N(\lambda_1) \leq Nh = w_2^N(\lambda_2)$. If job 1 chooses to wait in system 2 then so will job $i > 1$, because $w_i^N(\lambda_2) \geq w_1^N(\lambda_2) = Nh$, and $w_i^N(\lambda_2) = Nh$ if i waits. Again, because job i can also wait in system 1, we have $w_i^N(\lambda_1) \leq Nh = w_i^N(\lambda_2)$.

Now suppose job 1 uses a server in system 2; label that server A and label the other server B. Then

$$\begin{aligned} w_1^N(\lambda_2) &= h + \beta_A + \lambda_2 w_2^{N-1}(\lambda_2) + \mu_B w_1^{N-1}(\lambda_2) \\ &= h + \beta_A + \lambda_1 w_2^{N-1}(\lambda_2) + (\lambda_2 - \lambda_1) w_2^{N-1}(\lambda_2) + \mu_B w_1^{N-1}(\lambda_2) \\ &\geq h + \beta_A + \lambda_1 w_2^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1) w_1^{N-1}(\lambda_1) + \mu_B w_1^{N-1}(\lambda_1) \qquad (43) \end{aligned}$$

from the induction hypothesis and (42). Because we can also assign job 1 to server A in system 1,

$$w_1^N(\lambda_1) \leq h + \beta_A + \lambda_1 w_2^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1 + \mu_B) w_1^{N-1}(\lambda_1) \leq w_1^N(\lambda_2). \qquad (44)$$

Now consider job i (when job 1 uses server A in system 2). We have the following possible cases.

- Server B is offered to job i in system 2, and job i uses it. Let job i also use it in system 1. Note that this is possible, because job i will also be offered server B in system 1, from the induction hypothesis on N: $w_{i-1}^{N-1}(\lambda_1) \leq w_{i-1}^{N-1}(\lambda_2)$, and job $i-1$ will reject server B if $\beta_B/\mu_B > w_{i-1}^{N-1}(\lambda)$. Then, again using induction and (42), with calculations similar to those of (43) and (44), we have $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$.
- Server B is offered to job i in system 2, and job i does not use it. Again, job i will also be offered server B in system 1. Letting job i not use it in system 1, we have

$$w_i^N(\lambda_2) = h + \mu_A w_{i-1}^{N-1}(\lambda_2) + \mu_B w_i^{N-1}(\lambda_2) + \lambda_2 w_{i+1}^{N-1}(\lambda_2) \qquad (45)$$

$$w_i^N(\lambda_1) \leq h + \mu_A w_{i-1}^{N-1}(\lambda_1) + \mu_B w_i^{N-1}(\lambda_1) + \lambda_1 w_{i+1}^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1) w_i^{N-1}(\lambda_1), \qquad (46)$$

  and again, by induction and (42), $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$.
- Server B is not offered to job i in system 2, so both servers are busy in system 2. If job i is not offered server B in system 1, then the result follows as in the previous cases. If job i is offered

server B in system 1, we let job i use it until the next event. Then, because job i would have preferred to use server B in system 2 (because some higher priority job preferred it over waiting),

$$w_i^N(\lambda_2) \geq h + \beta_B + \mu_A w_{i-1}^{N-1}(\lambda_2) + \lambda_2 w_{i+1}^{N-1}(\lambda_2) \tag{47}$$

$$w_i^N(\lambda_1) \leq h + \beta_B + \mu_A w_{i-1}^{N-1}(\lambda_1) + \lambda_1 w_{i+1}^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1) w_i^{N-1}(\lambda_1), \tag{48}$$

and the result follows by induction and (42).

We have shown, for all N and i, that $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$. Taking limits yields $v_i(\lambda_1) \leq v_i(\lambda_2)$.

Proof of Theorem 28

If $\mu_1 \geq \mu_2$ the result holds from Theorem 26, so suppose $\mu_1 < \mu_2$. As in the proof of Theorem 26, server 1 will be preferred if and only if

$$\lambda((\mu_2 - \mu_1)v_2(\lambda) + (\beta_1 - \beta_2)) \leq h\mu_1 + \beta_2\mu_1 - h\mu_2 - \beta_1\mu_2. \tag{49}$$

The left hand side is strictly increasing in λ by Lemma 27, and the result follows.

Proof of Corollary 29

If server 1 is preferred for all λ, the result follows from Lemma 27. Suppose there exists $\lambda_0 < \infty$ such that server 1 is preferred if and only if $\lambda < \lambda_0$, so $\mu_2 > \mu_1$. Choose $\lambda_1 < \lambda_2 < \mu_1 + \mu_2$. If $\lambda_2 < \lambda_0$ or $\lambda_0 < \lambda_1$, then the result follows again from Lemma 27. Suppose $\lambda_1 < \lambda_0 < \lambda_2$. Then we have

$$v_{T(\lambda_1)}(\lambda_2) \geq v_{T(\lambda_1)}(\lambda_1) \geq \frac{\beta_2}{\mu_2} \geq \frac{\beta_1}{\mu_1} \tag{50}$$

where the first inequality follows from Lemma 27 and the last inequality follows from $\frac{h+\beta_1}{\mu_1} \leq \frac{h+\beta_2}{\mu_2}$ and $\mu_2 > \mu_1$. Hence we have $T(\lambda_2) \leq T(\lambda_1)$ from (32).

Proof of Lemma 30

Job $i = 1$ will start using server A, and if there is an arrival before it finishes service, it will become job 2. Hence, $v_1 = \frac{h+\beta_A+\lambda v_2}{\mu_A+\lambda}$.

For $i < T$, job i will become job $i+1$ ($i-1$) if the next transition is an arrival (departure). Therefore, this process is a gambler's ruin process, where the gambler starts with $i-1$ coins, wins each game with probability $p = \frac{\lambda}{\lambda+\mu_A}$, and eventually either loses everything (becomes job 1 in our setting) or ends up with $T-1$ coins (becomes job T in our setting). The probability of the latter is P_i given in (34).

For $1 < i < T$, the holding cost job i will pay until either it starts at server A or it becomes job T will be $H_i = hE[N_{i-1}^{T-1}]\frac{1}{\mu_A+\lambda}$ where $E[N_{i-1}^{T-1}] = \frac{i-1}{1-2p} - \frac{T-1}{1-2p}P_i$ is the expected number of games played in the gambler's ruin problem. Hence $v_i = H_i + P_i v_T + (1-P_i)v_1$.

Job $i = T$ will start using server B, and if there is an arrival (departure) before it finishes service, then it will become job $T+1$ ($T-1$). Hence $v_T = \frac{h+\beta_B+\mu_A v_{T-1}+\lambda v_{T+1}}{\mu_A+\mu_B+\lambda}$.

For $i > T$, the time until job i becomes job T will be stochastically identical to the sum of $i-T$ busy cycles in an $M/M/1$ queue with service rate $\mu_A+\mu_B$ and arrival rate λ, so $v_i = \frac{(i-T)h}{\mu_A+\mu_B-\lambda} + v_T$.

PROOF OF LEMMA 35

The proof is by induction on N. For $N = 1$, $w_i^1(\lambda) = h$, $x_i^1(\lambda) = h + \beta_B$ for all i and the result follows. Suppose the lemma is true for $N-1$. As noted before, we have $x_i^m(\lambda) - w_i^m(\lambda) = 0$ for all $i \geq T^m(\lambda)$. For $1 < i < T^m(\lambda)$, we have

$$x_i^N(\lambda) = h + \beta_B + \lambda x_{i+1}^{N-1}(\lambda) + \mu_A x_{i-1}^{N-1}(\lambda) \tag{51}$$

$$w_i^N(\lambda) = h + \lambda w_{i+1}^{N-1}(\lambda) + \mu_A w_{i-1}^{N-1}(\lambda) + \mu_B w_i^{N-1}(\lambda). \tag{52}$$

Hence,

$$\begin{aligned} x_i^N(\lambda) - w_i^N(\lambda) &= \beta_B + \lambda[x_{i+1}^{N-1}(\lambda) - w_{i+1}^{N-1}(\lambda)] \\ &\quad + \mu_A[x_{i-1}^{N-1}(\lambda) - w_{i-1}^{N-1}(\lambda)] - \mu_B w_i^{N-1}(\lambda) \qquad (53) \\ x_{i+1}^N(\lambda) - w_{i+1}^N(\lambda) &= \beta_B + \lambda[x_{i+2}^{N-1}(\lambda) - w_{i+2}^{N-1}(\lambda)] \\ &\quad + \mu_A[x_i^{N-1}(\lambda) - w_i^{N-1}(\lambda)] - \mu_B w_{i+1}^{N-1}(\lambda), \qquad (54) \end{aligned}$$

so the result follows from induction on N and Lemma 34. For $i = 1$,

$$x_1^N(\lambda) = h + \beta_B + \lambda x_2^{N-1}(\lambda) + \mu_A x_1^{N-1}(\lambda) \tag{55}$$

$$w_1^N(\lambda) \leq h + \lambda w_2^{N-1}(\lambda) + (\mu_A+\mu_B) w_1^{N-1}(\lambda), \tag{56}$$

where the right hand side for the inequality is the cost incurred until the next decision epoch when the highest priority job does not use server A. Hence,

$$x_1^N(\lambda) - w_1^N(\lambda) \geq \beta_B + \lambda[x_2^{N-1}(\lambda) - w_2^{N-1}(\lambda)] + \mu_A[x_1^{N-1}(\lambda) - w_1^{N-1}(\lambda)] - \mu_B w_1^{N-1}(\lambda), \tag{57}$$

and the result follows from induction on N and Lemma 34.

Proof of Lemma 36

Choose $\lambda_1, \lambda_2 \in \mathcal{L}$ such that $\lambda_1 < \lambda_2 < \mu_1 + \mu_2$, and rescale so that $\lambda_2 + \mu_1 + \mu_2 = 1$. Again we use induction on N. We modify our model to permit jobs to leave server A and rejoin the queue at any time. In the limit, as $N \to \infty$, this option will never be exercised (it increases holding costs without reducing operating costs). For $N = 1$ and for all λ, $w_i^1(\lambda) = h$ and $x_i^1(\lambda) = h + \beta_B$ for all i, hence (i) and (ii) are true. Suppose $w_i^{N-1}(\lambda_1) \leq w_i^{N-1}(\lambda_2)$, $x_i^{N-1}(\lambda_1) \leq x_i^{N-1}(\lambda_2)$ and $x_i^{N-1}(\lambda_1) - w_i^{N-1}(\lambda_1) \geq x_i^{N-1}(\lambda_2) - w_i^{N-1}(\lambda_2)$ for all i. Now we show $x_i^N(\lambda_1) - w_i^N(\lambda_1) \geq x_i^N(\lambda_2) - w_i^N(\lambda_2)$ for all i. Suppose the highest priority job in system 2 (with rate λ_2), prefers to use server A rather than idling. We let it do the same in system 1 (with rate λ_1). Then,

$$\begin{aligned} x_1^N(\lambda_1) - w_1^N(\lambda_1) \geq\ & \lambda_1[x_2^{N-1}(\lambda_1) - w_2^{N-1}(\lambda_1)] + \mu_A[x_1^{N-1}(\lambda_1) - w_1^{N-1}(\lambda_1)] \\ & + (\lambda_2 - \lambda_1)[x_1^{N-1}(\lambda_1) - w_1^{N-1}(\lambda_1)] - (\mu_B - \mu_A)w_1^{N-1}(\lambda_1) \\ & + \beta_B - \beta_A \end{aligned} \tag{58}$$

$$\begin{aligned} x_1^N(\lambda_2) - w_1^N(\lambda_2) =\ & \lambda_1[x_2^{N-1}(\lambda_2) - w_2^{N-1}(\lambda_2)] + \mu_A[x_1^{N-1}(\lambda_2) - w_1^{N-1}(\lambda_2)] \\ & + (\lambda_2 - \lambda_1)[x_2^{N-1}(\lambda_2) - w_2^{N-1}(\lambda_2)] - (\mu_B - \mu_A)w_1^{N-1}(\lambda_2) \\ & + \beta_B - \beta_A, \end{aligned} \tag{59}$$

and the result follows by induction and Lemma 35. On the other hand if job 1 in system 2 (with rate λ_2), prefers to idle server A, we let it do the same in system 1 (with rate λ_1). Then,

$$\begin{aligned} x_1^N(\lambda_1) - w_1^N(\lambda_1) \geq\ & \beta_B + \lambda_1[x_2^{N-1}(\lambda_1) - w_2^{N-1}(\lambda_1)] + \mu_A[x_1^{N-1}(\lambda_1) - w_1^{N-1}(\lambda_1)] \\ & + (\lambda_2 - \lambda_1)[x_1^{N-1}(\lambda_1) - w_1^{N-1}(\lambda_1)] - \mu_B w_1^{N-1}(\lambda_1) \end{aligned} \tag{60}$$

$$\begin{aligned} x_1^N(\lambda_2) - w_1^N(\lambda_2) =\ & \beta_B + \lambda_1[x_2^{N-1}(\lambda_2) - w_2^{N-1}(\lambda_2)] + \mu_A[x_1^{N-1}(\lambda_2) - w_1^{N-1}(\lambda_2)] \\ & + (\lambda_2 - \lambda_1)[x_2^{N-1}(\lambda_2) - w_2^{N-1}(\lambda_2)] - \mu_B w_1^{N-1}(\lambda_2), \end{aligned} \tag{61}$$

and the result follows by induction and Lemma 35. Thus we have shown $x_i^N(\lambda_1) - w_i^N(\lambda_1) \geq x_i^N(\lambda_2) - w_i^N(\lambda_2)$ for $i = 1$.

From Equation (37), the induction hypothesis, and Lemma 35, we have $T^N(\lambda_2) \leq t = T^N(\lambda_1)$. For

$1 < i < T^N(\lambda_2)$, we have

$$\begin{aligned} x_i^N(\lambda_1) - w_i^N(\lambda_1) &= \beta_B + \lambda_1[x_{i+1}^{N-1}(\lambda_1) - w_{i+1}^{N-1}(\lambda_1)] + \mu_A[x_{i-1}^{N-1}(\lambda_1) - w_{i-1}^{N-1}(\lambda_1)] \\ &\quad + (\lambda_2 - \lambda_1)[x_i^{N-1}(\lambda_1) - w_i^{N-1}(\lambda_1)] - \mu_B w_i^{N-1}(\lambda_1) \qquad (62) \\ x_i^N(\lambda_2) - w_i^N(\lambda_2) &= \beta_B + \lambda_1[x_{i+1}^{N-1}(\lambda_2) - w_{i+1}^{N-1}(\lambda_2)] + \mu_A[x_{i-1}^{N-1}(\lambda_2) - w_{i-1}^{N-1}(\lambda_2)] \\ &\quad + (\lambda_2 - \lambda_1)[x_{i+1}^{N-1}(\lambda_2) - w_{i+1}^{N-1}(\lambda_2)] - \mu_B w_i^{N-1}(\lambda_2), \qquad (63) \end{aligned}$$

and the result follows by induction and Lemma 35. For $T^m(\lambda_2) \leq i < T^m(\lambda_1)$, we have $x_i^N(\lambda_1) - w_i^N(\lambda_1) \geq 0 = x_i^N(\lambda_2) - w_i^N(\lambda_2)$, and for $i \geq T^m(\lambda_1)$ we have $x_i^N(\lambda_1) - w_i^N(\lambda_1) = 0 = x_i^N(\lambda_2) - w_i^N(\lambda_2)$. Thus, we have shown $x_i^N(\lambda_1) - w_i^N(\lambda_1) \geq x_i^N(\lambda_2) - w_i^N(\lambda_2)$ for all i.

Next we show $w_i^N(\lambda_1) \leq w_i^N(\lambda_2)$ for all i. First we consider $i = 1$. If the job prefers to use server A rather than idle in system 2, we let it do the same in system 1, and we have

$$w_1^N(\lambda_2) = h + \beta_A + \lambda_2 w_2^{N-1}(\lambda_2) + \mu_B w_1^{N-1}(\lambda_2) \qquad (64)$$

$$w_1^N(\lambda_1) \leq h + \beta_A + \lambda_1 w_2^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1 + \mu_B) w_1^{N-1}(\lambda_1), \qquad (65)$$

and the result follows by induction and Lemma 34. If the job prefers to idle in system 2, then we let it do the same in system 1, and the result follows similarly. Next we consider $i > 1$. We look at the following possible cases.

- $i = T^N(\lambda_2) \leq T^N(\lambda_1)$: Let job i also use server B in system 1, by preempting the lower priority job. Then

$$w_i^N(\lambda_2) = h + \beta_B + \mu_A x_{i-1}^{N-1}(\lambda_2) + \lambda_1 x_{i+1}^{N-1}(\lambda_2) + (\lambda_2 - \lambda_1) x_{i+1}^{N-1}(\lambda_2) \qquad (66)$$

$$w_i^N(\lambda_1) \leq h + \beta_B + \mu_A x_{i-1}^{N-1}(\lambda_1) + \lambda_1 x_{i+1}^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1) x_i^{N-1}(\lambda_1). \qquad (67)$$

- $i < T^N(\lambda_2) \leq T^N(\lambda_1)$: We have

$$w_i^N(\lambda_2) = h + \mu_A w_{i-1}^{N-1}(\lambda_2) + \lambda_2 w_{i+1}^{N-1}(\lambda_2) + \mu_B w_i^{N-1}(\lambda_2) \qquad (68)$$

$$w_i^N(\lambda_1) = h + \mu_A w_{i-1}^{N-1}(\lambda_1) + \lambda_1 w_{i+1}^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1 + \mu_B) w_i^{N-1}(\lambda_1). \qquad (69)$$

- $T^N(\lambda_2) \leq T^N(\lambda_1) < i$: We have

$$w_i^N(\lambda_2) = h + (\mu_A + \mu_B) w_{i-1}^{N-1}(\lambda_2) + \lambda_2 w_{i+1}^{N-1}(\lambda_2) \qquad (70)$$

$$w_i^N(\lambda_1) = h + (\mu_A + \mu_B) w_{i-1}^{N-1}(\lambda_1) + \lambda_1 w_{i+1}^{N-1}(\lambda_1) + (\lambda_2 - \lambda_1) w_i^{N-1}(\lambda_1). \qquad (71)$$

- $T^N(\lambda_2) < i \leq T^N(\lambda_1)$: We let job i use server B in system 1, by preempting the lower priority job in system 1. Then, because job i would have preferred to use server B in system 2 (because some higher priority job preferred it over waiting),

$$w_i^N(\lambda_2) \geq h + \beta_B + \mu_A x_{i-1}^{N-1}(\lambda_2) + \lambda_2 x_{i+1}^{N+1}(\lambda_2) \tag{72}$$

$$w_i^N(\lambda_1) \leq h + \beta_B + \mu_A x_{i-1}^{N-1}(\lambda_1) + \lambda_1 x_{i+1}^{N+1}(\lambda_1) + (\lambda_2 - \lambda_1) x_i^{N-1}(\lambda_1). \tag{73}$$

In all cases the result follows by induction and Lemma 34.

Finally, similar arguments yield $x_i^N(\lambda_1) \leq x_i^N(\lambda_2)$ and the proof is complete.

Proofs of Corollary 33 and Theorem 32

The proof of Corollary 33 is contained in the previous proof of Lemma 36. The proof of Theorem 32 is similar to the proof of Theorem 28, using Lemmas 34, 35 and 36 above.

References

[1] U.S. Environmental Protection Agency Energy Star Program. (2007, Aug.) Report to congress on server and data center energy efficiency public law 109-431. [Online]. Available: http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf

[2] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, pp. 1458–1472, Nov. 2008.

[3] M. Armony and A. Ward, "Fair dynamic routing in large-scale heterogeneous-server systems," *Operat. Res.*, vol. 58, pp. 624–637, 2010.

[4] P. Naor, "On the regulation of queue size by levying tolls," *Econometrica*, vol. 37, pp. 15–24, 1969.

[5] U. Yechiali, "Customers' optimal joining rules for the GI/M/s queue," *Management Science*, vol. 18, pp. 434–443, 1972.

[6] S. Stidham, "Optimal control of admission to a queueing system," *IEEE Trans. Autom. Control*, vol. AC-30, pp. 705–713, Aug. 1985.

[7] V. G. Kulkarni and N. Gautam, "Admission control of multi-class traffic with service priorities in high-speed networks," *Queueing Systems*, vol. 27, pp. 1–16, 1997.

[8] S. H. Xu and J. G. Shantikumar, "Optimal expulsion control - a dual approach to admission control of an ordered-entry system," *Operat. Res.*, vol. 41, pp. 1137–1152, 1993.

[9] G. Weiss and M. Pinedo, "Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions," *Journal of Applied probability*, vol. 17, pp. 187–202, 1980.

[10] A. K. Agrawala, E. G. J. Coffman, M. R. Garey, and S. K. Tripathi, "A stochastic optimization algorithm minimizing expected flow times on uniform processors," *IEEE Trans. Comput.*, vol. C-33, pp. 351–356, Apr. 1984.

[11] P. Lin and P. R. Kumar, "Optimal control of a queueing system with two heterogeneous servers," *IEEE Trans. Autom. Control*, vol. AC-29, pp. 696–703, Aug. 1984.

[12] G. Koole, “A simple proof of the optimality of a threshold policy in a two-server queueing system,” *Systems and Control Letters*, vol. 26, pp. 301–303, 1995.

[13] J. Walrand, “A note on “optimal control of a queueing system with two heterogeneous servers”,” *Systems and Control Letters*, vol. 4, pp. 131–134, 1984.

[14] S. H. Xu, “A duality approach to admission and scheduling control of queues,” *Queueing Systems*, vol. 18, pp. 273–300, 1994.

[15] R. H. Stockbridge, “A martingale approach to the slow server problem,” *Journal of Applied Probability*, vol. 28, pp. 480–486, 1991.

[16] I. Viniotis and A. Ephremides, “Extension of optimality of the threshold policy in heterogeneous multiserver queueing systems,” *IEEE Trans. Autom. Control*, vol. 33, pp. 104–109, Sep. 1988.

[17] V. V. Rykov, “Monotone control of queueing systems with heterogeneous servers,” *Queueing Systems*, vol. 37, pp. 391–403, 2001.

[18] F. de Vèricourt and Y. P. Zhou, “On the incomplete results for the heterogeneous server problem,” *Queueing Systems*, vol. 52, pp. 189–191, 2006.

[19] H. Luh and I. Viniotis, “Threshold control policies for heterogeneous server systems,” *Mathematical Methods of Operations Research*, vol. 55, pp. 121–142, 2002.

[20] Z. Rosberg and A. M. Makowski, “Optimal routing to parallel heterogeneous servers-small arrival rates,” *IEEE Trans. Autom. Control*, vol. 35, pp. 789–796, Jul. 1990.

[21] R. R. Weber, “On a conjecture about assigning jobs to processors of different speeds,” *IEEE Trans. Autom. Control*, vol. 38, pp. 166–170, Jan. 1993.

[22] J. H. Kim, H. S. Ahn, and R. Righter, “Managing queues with heterogeneous servers,” *Journal of Applied Probability*, vol. 48, pp. 435–452, 1999.

[23] C. E. Bell, “Optimal operation of an M/M/2 queue with removable servers,” *Operat. Res.*, vol. 28, pp. 1189–1204, 1980.

[24] O. C. Ibe and J. Keilson, “Multi-server threshold queues with hysteresis,” *Performance Evaluation*, vol. 21, pp. 185–213, 1995.

[25] J. C. S. Liu and L. Golubchik, “Stochastic complement analysis of multi-server threshold queues with hysteresis,” *Performance Evaluation*, vol. 35, pp. 19–48, 1999.

[26] R. Zhang and Y. A. Phillis, “Fuzzy control of queueing systems with heterogeneous servers,” *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 17–26, Feb. 1999.

[27] L. L. H. Andrew, M. Lin, and A. Wierman, “Optimality, fairness, and robustness in speed scaling designs,” in *Proc. ACM Sigmetrics*, 2010.

[28] A. Wierman, L. L. H. Andrew, and A. Tang, “Power-aware speed scaling in processor sharing systems,” in *Proc. IEEE INFOCOM*, Apr. 2009.

[29] V. V. Rykov and D. V. Efrosinin, “On the slow server problem,” *Automation and Remote Control*, vol. 70, pp. 2013–2023, 2009.

[30] R. Hassin, “On the optimality of first-come last-served queues,” *Econometrica*, vol. 53, pp. 201–202, 1985.

[31] R. Righter and S. H. Xu, “Scheduling jobs on non-identical ifr processors to minimize genral cost functions,” *Advances in Applied Probability*, vol. 23, pp. 909–924, 1991.

[32] E. G. J. Coffman, L. Flatto, M. R. Garey, and R. R. Weber, “Minimizing expected makespans on uniform processor systems,” *Advances in Applied Probability*, vol. 19, pp. 177–201, 1987.

[33] P. R. Kumar and J. Walrand, “Individually optimal routing in parallel systems,” *Journal of Applied Probability*, vol. 22, pp. 989–995, 1986.

[34] S. M. Ross, *Introduction to Probability Models*. Burlington, MA: Academic Press, 2010.