An Optimal Service Ordering for a World Wide Web Server*

Amy Csizmar Dalal
Hewlett-Packard Laboratories
amy_dalal@hp.com

Scott Jordan
University of California at Irvine
sjordan@uci.edu

Abstract

We consider alternative service policies in a web server with impatient users. User-perceived performance is modeled as an exponentially decaying function of the user's waiting time, reflecting the probability that the user aborts the download before the page is completely received. The web server is modeled as a single server queue, with Poisson arrivals and exponentially distributed file lengths. The server objective is to maximize average revenue per unit time, where each user is assumed to pay a reward proportional to the perceived performance. When file lengths are i.i.d., we prove that the optimal service policy is greedy, namely that the server should choose the job with the highest potential reward. However, when file lengths are independently drawn from a set of exponential distributions, we show the optimal policy need not be greedy; in fact, processor sharing policies sometimes outperform the best greedy policy in this case.

1 Introduction

Most of today's web servers handle incoming web requests in a round robin manner. This method of scheduling requests is partly a consequence of operating system design and partly due to the web server software design. Traditionally, it has been held that processor sharing policies such as round robin yield the best "user-perceived" performance, in terms of fairness and response time, when service time requirements are variable. Processor sharing policies typically allow shorter jobs to complete before longer jobs, and the average time a job spends in a system that employs processor sharing is linearly proportional to its required service time.

Web users tend to be very impatient and will abort a pending request if a response is not received within seconds. Users that time out in such a manner cause the server to waste resources on a request that never completes. At a heavily-loaded server with many requests arriving per second, this may lead to a situation of "server deadlock", where the server works at capacity without completing any requests.

We use a queueing theory approach to derive an optimal service ordering policy for a system consisting of a web server processing requests for files, or web documents, from remote users. The objective of the web server in the system studied here is to maximize user perceived performance, which is a decreasing function of the amount of time the user spends waiting for a file to download from a web server. We assume that the network connecting the client and server is a static quantity and look solely at what the server does in processing requests. Our metric of interest is a quantity we call "revenue", defined as the probability that the user has not aborted a request before it is filled by the server. We describe revenue as a decaying exponential function of the time the request spends in the system before completing service. Our goal is to find the service policy that maximizes the average revenue earned per second by the server.

Several previous research efforts have utilized similar queueing methods to analyze quality of service issues in web servers. Both [2] and [5], for instance, demonstrate improvements in response time at a web server by using a non-traditional service ordering (shortest-connection-first and shortest-remaining-processing-time, respectively) in place of processor sharing. [6] uses a queueing model to determine how response time at a web server is affected by several parameters, including network bandwidth, processor speed, and adding additional hosts to a distributed server system. However, [6] only considers first-in, first-out service ordering, and does not study how alternate policies may affect response time. [1] proposed an improvement over FIFO service at web servers by using a combination of admission control and a set of priority queues, thus providing better QoS at the web server and increasing the throughput of a web server as seen by the clients. However, none of these studies considers the "impatient user" problem.

The rest of this paper is organized as follows. In Section 2, we describe the analytical model used to represent the web server system. In Section 3, we derive the policy which will maximize the server's average revenue per unit time for this model. Section 4 discusses the derivations of the optimal policies for two variations to the original model and presents a counterexample which limits the optimal policies we can consider for one of these extensions. Finally, we present our conclusions in Section 5.

2 System model

Our model is an abstraction of the actions that occur at the session layer due to the HTTP/1.1 protocol [4]. We ignore all actions associated with the network layer, including specifics about individual TCP connections associated with requests. We

*This paper represents research done while both authors were at Northwestern University.

assume the server is not aware if or when a request is aborted before it is processed to completion.

We model the web server as a single-server queue with a single stream of Poisson arrivals. The service times of incoming requests are drawn from an exponential distribution with parameter μ; service times are independent and identically distributed. The service time of a request is proportional to the number of bytes in the requested file. Further, we assume that swap times are instantaneous.

The web server system alternates between idle cycles, when no requests are awaiting service, and busy cycles, when at least one request is awaiting service. Busy and idle cycles are i.i.d., and the two are independent.

We denote the arrival time of request i as x_i and the departure time of request i under some service policy p as t_{i_p}. The *response time* of request i under policy p is $t_{i_p} - x_i$.

The server earns an amount of revenue from serving request i under policy p equal to $g_{i_p}(t_{i_p}) = e^{-c(t_{i_p}-x_i)}$. The server collects this amount once request i departs the system. Additionally, we define the *potential revenue* for request i at a time t prior to its departure time as $g_{i_p}(t) = e^{-c(t-x_i)}$.

Let N denote the number of jobs served in the first busy cycle. By the Renewal Reward Theorem, the average reward per unit time earned by the server under policy p is $V_p(\emptyset) = E[\sum_{i=1}^{N} g_{i_p}(t_{i_p})]/E[Z_1]$, where $\emptyset$ indicates that the policy starts during an idle cycle and Z_1 is the length of the first busy/idle cycle. The optimal server policy satisfies $V = \max_p V_p(\emptyset)$.

The policy space under consideration, $\mathcal{S}$, includes all preemptive and non-preemptive policies, policies that work on one request at a time, policies that work on multiple requests simultaneously, and both idling and non-idling policies.

3 Derivation of the optimal service ordering policy

Given the above assumptions, we can determine an optimal service ordering policy from the set of the policies in $\mathcal{S}$. Such a policy must satisfy a set of requirements which we define below. We then derive the optimal policy from the set of remaining policies that satisfy these requirements.

Lemma 1 *An optimal policy can be found among the class of non-idling policies.*

Proof: We show that this lemma holds using a proof by contradiction, which we briefly sketch here. The full proof for this and all other proofs presented in this paper can be found in [3]. Suppose there exists an optimal policy p which is optimal, yet idles over a time interval $[a_1, a_2)$ of a sample path in the first busy cycle while requests are queued. We show that by taking an infinitesimally small time interval of size dl, where dl is small enough that at most one departure can occur with probability $\mu\, dl + o(dl)$, from $[a_2, Z_1)$ and swapping this interval with an interval of size dl in $[a_1, a_2)$, the expected revenue improves by some nontrivial positive quantity, establishing a contradiction. By repeating this method, we in fact show that we can continue improving the expected revenue until the idle period lies completely after all requests are serviced. □

Lemma 2 *An optimal service ordering can be found in the class of policies that switch between jobs in service only upon an arrival to or departure from the system.*

Proof: We prove this lemma by contradiction. We present the case where the policies are non-processor-sharing; the processor sharing case follows directly.

Suppose there exists an optimal policy p which switches between jobs in service at some time other than a departure time or arrival time over some interval of a sample path in the first busy cycle. At some time interval $[a_1 - dl, a_1)$ during this busy cycle, the server works on job k. At time a_1, the server switches to job $(k+1)$, and processes that job for at least the interval $[a_1, a_1 + dl)$. No arrivals or departures occur at time a_1. At some future time a_2, the server switches back to serving job k. We make no assumptions as to the jobs served in the interval $[a_1 + dl, a_2)$.

We construct a policy p' which is identical to policy p in the intervals $[0, a_1 - dl)$, $[a_1 + dl, a_2)$, and $[a_2 + dl, \infty)$. Our construction of p' depends on the potential revenues of jobs k and $(k+1)$ at time a_1, denoted as $g_{k_p}(a_1)$ and $g_{(k+1)_p}(a_1)$, respectively.

If $g_{(k+1)_p}(a_1) > g_{k_p}(a_1)$, then we construct p' as follows: In the interval $[a_1 - dl, a_1)$, serve job $(k+1)$. At a_1, switch to job k and serve k in the interval $[a_1, a_1 + dl)$. Then, serve job k in the interval $[a_2, a_2 + dl)$. In this case, p' differs from p only over $[a_1 - dl, a_1 + dl)$. The difference in the expected revenue under the two policies is $E[g_{(k+1)_{p'}}(a_1) + g_{k_{p'}}(a_1 + dl)] - E[g_{k_p}(a_1) + g_{(k+1)_p}(a_1 + dl)]$. By definition, there is no departure at time a_1 under policy p, but there may be a departure at a_1 under policy p'. Also, job k cannot depart the system prior to time $a_2 + dl$. The difference in expected revenues is thus $\mu dl\, e^{-c(a_1 - x_{(k+1)})}(1 - e^{-c\,dl})$, which is a positive quantity.

If, however, $g_{(k+1)_p}(a_1) < g_{k_p}(a_1)$, then we construct p' such that p' serves job k in the interval $[a_1 - dl, a_1 + dl)$ and serves job $(k+1)$ in the interval $[a_2, a_2 + dl)$. Here, p' differs from p over the intervals $[a_1, a_1 + dl)$ and $[a_2, a_2 + dl)$. The difference in the expected revenue under the two policies is $E[g_{k_{p'}}(a_1 + dl) + g_{(k+1)_{p'}}(a_2 + dl)] - E[g_{(k+1)_p}(a_1 + dl) + g_{k_p}(a_2 + dl)]$. There is no departure at time a_1 under policy p, but there may be a departure at either $a_1 + dl$ or $a_2 + dl$ (or both) under policy p'. The expected revenue is thus $\mu dl e^{-c\,dl}(e^{-ca_1} - e^{-ca_2})(e^{cx_k} - e^{cx_{(k+1)}})$. Since $g_{(k+1)_p}(a_1) < g_{k_p}(a_1)$, x_k must be greater than $x_{(k+1)}$, and the last term in this equation is positive.

Thus, for each policy p', we have shown that with some nonzero probability, $V_{p'}(\emptyset) > V_p(\emptyset)$, and therefore p cannot be an optimal policy. □

Lemma 3 *An optimal policy can be found in the class of non-processor-sharing policies.*

Proof: We consider the two-job case; the n-job case follows directly, since service times are independent. Suppose there exists a generalized processor-sharing (PS) policy that generates a higher revenue than the best non-processor-sharing (NPS) policy. Then, over some interval $[a, b]$, the server splits its resources between two jobs. Prior to time a, job 1 has spent an amount of time equal to τ_1 in the system, and job 2 has spent a time equal to τ_2 in the system. After time a, the distribution for the total revenue that the server earns from serving the two jobs under policy PS is $g_{\text{PS}}(t) = g_{1_{\text{PS}}}(t) + g_{2_{\text{PS}}}(t) = c_f e^{-cT_f} + c_s e^{-cT_s}$, where

- $T_f = \min(T_1', T_2')$, with T_1' and T_2' defined as the remaining service times of jobs 1 and 2, respectively. T_f is exponentially distributed with parameter μ.
- $T_s = T_f + Z$, where Z is an exponentially-distributed random variable with parameter μ.
- If $T_f = T_1'$, then $c_f = e^{-c\tau_1} \equiv c_1$ and $c_s = e^{-c\tau_2} \equiv c_2$. Otherwise, $c_f = e^{-c\tau 2} \equiv c_2$ and $c_s = e^{-c\tau 1} \equiv c_1$.

The expected value of the revenue gained over this portion of the sample path is

$$E[g_{\text{PS}}(t)] = \int_0^\infty \int_0^\infty (c_f e^{-ct_f} + c_s e^{-ct_s}) f_{T_f,T_s}(t_f, t_s)\, dt_f\, dt_s$$

where $f_{T_f,T_s}(t_f, t_s)$ is the joint density of the service times of the two jobs, given by $f_{T_f,T_s}(t_f, t_s) = \mu^2 e^{-\mu(t_s - t_f)} e^{-\mu t_f}$. Substituting $z = t_s - t_f$ and integrating yields $E[g_{\text{PS}}(t)] = \mu/(c+\mu)\, c_f + (\mu/(c+\mu))^2 c_s$.

There are two possible values for c_f and c_s, depending on which job departs the system first. Thus, if job 1 departs the system first with probability q, the expected revenue from policy PS is

$$\begin{aligned} E[g_{\text{PS}}(t)] &= q\left[\tfrac{\mu}{c+\mu} c_1 + \left(\tfrac{\mu}{c+\mu}\right)^2 c_2\right] \\ &\quad + (1-q)\left[\tfrac{\mu}{c+\mu} c_2 + \left(\tfrac{\mu}{c+\mu}\right)^2 c_1\right] \end{aligned} \quad (1)$$

If, instead, the server chooses to work on one job first and then the other, there are two possible policies. Define policy NPS1 as the policy that serves job 1 to completion and then serves job 2, and policy NPS2 as the policy that serves job 2 to completion and then serves job 1.

Under NPS1, the total potential revenue is $g_{\text{NPS1}}(t) = c_1 e^{-cT_1'} + c_2 e^{-c(T_1' + T_2')}$, and under NPS2 it is $g_{\text{NPS2}}(t) = c_1 e^{-c(T_1' + T_2')} + c_2 e^{-cT_2'}$, where T_1', T_2', τ_1, τ_2, c_1, and c_2 are as defined previously.

The expected revenue under policy NPS1 over this interval is

$$\begin{aligned} E[g_{\text{NPS1}}(t)] &= \int_0^\infty \int_0^\infty e^{-ct_1'}(c_1 + c_2 e^{-ct_2'}) \\ &\qquad \mu^2 e^{-\mu t_1'} e^{-\mu t_2'}\, dt_1'\, dt_2' \\ &= \tfrac{\mu}{c+\mu} c_1 + \left(\tfrac{\mu}{c+\mu}\right)^2 c_2 \end{aligned} \quad (2)$$

Similarly, the expected revenue under policy NPS2 is

$$E[g_{\text{NPS2}}(t)] = \left(\tfrac{\mu}{c+\mu}\right)^2 c_1 + \tfrac{\mu}{c+\mu} c_2 \quad (3)$$

However, (1) is just a weighted sum of (2) and (3). Therefore, (1) cannot be greater than both (2) and (3). Thus, policy PS cannot earn a strictly higher revenue than both policy NPS1 and policy NPS2.□

The above analysis assumes a fixed percentage of resources expended by the processor on each job under policy PS; the results hold for the variable percentage case as well.

Lemma 4 *An optimal policy can be found in the class of Markov policies; it is independent of both past and future arrivals.*

This lemma is a simple consequence of the exponential interarrival times and service times of the requests in the system. □

We now present two definitions which will aid in the derivation of the optimal policy. From Lemmas 1 and 3, we know that in each interval in which there is at least one job in the system, the server will select one job to process in that interval. The server bases its selection on its determination of which job will maximize its total revenue and makes its selection from the pool of jobs that have not completed service and are still in the system at time t. Since jobs that have already departed do not affect the service selection, the state of the system is defined completely by the arrival times of the jobs presently in the system:

Definition 1 *The* state vector *of the web server system at time t under a service policy p is given by*

$$G_{t_p} = \{g_{1_p}(t)\ \ g_{2_p}(t)\ \ \ldots\ \ g_{M_p}(t)\}$$

where M denotes the number of jobs that have arrived at the system since the start of the current busy cycle and have not departed prior to t, and $g_{i_p}(t) = e^{-c(t - x_i)}$.

Since each $g_{i_p}(t)$ is an exponential function, each element of the state vector will decay by the same ratio over each time interval, until the job departs the system, at which point its revenue function ceases to decay.

Because the distribution of remaining service time is identical for every job in the system at time t, the expected revenue gained from serving any one job i with remaining service time t_i' from time t until completion is $E[g_{i_p}(t_i')] = g_{i_p}(t)\, \mu/(c+\mu)$. The job with the highest expected payoff is $\arg\max_i \mu/(c+\mu)\, g_{i_p}(t) = \arg\max_i g_{i_p}(t), i \in 1, 2, \ldots, M$. We define this job as the "best job" in the system.

We now state the following theorem about the optimal policy:

Theorem 1 *An optimal policy is greedy. That is, it chooses the "best job" to serve at any time t, regardless of the order chosen for the other jobs in the system.*

Proof: Assume there exists an optimal policy p that is work-conserving, switches jobs in service only at an arrival or departure epoch, is non-processor-sharing and Markov, but not

greedy. That is, at some time t in a sample path during a busy cycle, the server chooses to serve job k at time t, complete that job, and then serve job $(k+1)$ to completion, where $g_{(k+1)_p}(t) > g_{k_p}(t)$.

The state vector at time t is given in Definition 1. We can reorder these values in terms of the order in which the jobs are served under policy p: $G_{t_p} = \{g_{k_p}(t)\ g_{(k+1)_p}(t)\ \ldots\ g_{M_p}(t)\}$. The revenue obtained from policy p from t until the end of the current busy cycle is $\sum_{j=k}^{M} g_{j_p}(t) e^{-c\sum_{l=k}^{j} t'_{l_p}}$, where t'_{l_p} is the remaining service time of the lth job served after time t under policy p.

We construct a policy p' that performs an interchange of jobs k and $(k+1)$, such that the server processes job $(k+1)$ at time t to completion and then processes job k to completion. The revenue earned under policy p' from t until the end of the current busy cycle is $g_{(k+1)_p}(t) e^{-ct'_{(k+1)_p}} + g_{k_p}(t) e^{-c(t'_{k_p}+t'_{(k+1)_p})} + \sum_{j=k+2}^{M} g_{j_p}(t) e^{-c\sum_{l=k}^{j} t'_{l_p}}$.

The remaining service times for job k and job $(k+1)$ are the same under policy p and policy p'. Since service times are identically distributed, we let t'_1 denote the service time of the *first* job served after time t in both policy p and policy p' and t'_2 denote the service time of the *second* job served after time t under both policies.

The difference in expected revenues between the two policies is $E[e^{-ct'_1}(1-e^{-ct'_2})(g_{(k+1)_p}(t) - g_{k_p}(t))]$. But the term inside the expected value operator is always positive, so its expected value will also always be positive. Therefore, since the expected time of the busy cycle is the same under policy p and policy p', we conclude that $V_{p'} > V_p$, and thus p cannot be an optimal policy. □

For the system we have defined here, with identical cost constants and identical service time distributions, the greedy policy defaults to a preemptive-resume last-in-first-out (LIFO-PR) service policy. To see why this is so, we apply the argument from the proof of Theorem 1. Clearly, the newest job has the highest potential revenue value in the state vector, and is thus the "best job".

4 Two extensions to the original model

In this section, we consider two permutations of the web server model presented previously. In the original model, all incoming requests initially have a probability of one of remaining at the server until they complete their required service times. In the first extension, we modify the model so that the initial probability, or *initial reward*, varies among the incoming requests. In the second extension, we consider the case where the file sizes of incoming requests are drawn from a family of exponential distributions, each with a unique mean. This case is analogous to a web server that hosts several different types of content files (HTML files, image files, CGI scripts, et cetera) that are best described by different exponential distributions.

4.1 Extension 1: Varying initial reward

In this system, each incoming request i is weighted by some initial reward C_i. The potential revenue for request i for this system under an arbitrary policy p is $C_i e^{-c(t-x_i)} = C_i g_{i_p}(t)$. This system is a natural extension of the original web server model. The derivation of the optimal policy is identical to the derivation presented in Section 3, with the exception of the definition of the state vector:

Definition 2 *The* state vector *of the web server system at time t under a service policy p is given by*

$$G_{t_p} = \{C_1\, g_{1_p}(t)\ \ C_2\, g_{2_p}(t)\ \ \ldots\ \ C_M\, g_{M_p}(t)\}$$

where M and $g_{i_p}(t)$ are as defined in Definition 1 and C_i is the initial reward of request i.

The "best job" is now $\arg\max_i C_i\, g_{i_p}(t)\, \mu/(c+\mu) = \arg\max_i C_i\, g_{i_p}(t)$, and the optimal policy can be stated as follows:

Theorem 2 *An optimal policy is greedy; it chooses the best job to serve at any time t regardless of the order chosen to process the rest of the jobs in the web server system, where the best job is the pending request with the highest $C_i g_{i_p}(t)$ value in the state vector.*

The proof is identical to the proof of Theorem 1, with the new definition of potential revenue to account for the differing initial rewards. In this system, the optimal policy processes requests as follows: At any time, the web server will choose to serve the request that is the most profitable combination of time-in-system and initial payoff.

4.2 Extension 2: Varying mean file size

In this section, we consider a system in which service times are independently drawn from a set of N exponential distributions. We define $\mathcal{S}'$ as the set of policies under consideration for this system. For reasons which will be explained shortly, $\mathcal{S}'$ is a subset of $\mathcal{S}$ that excludes policies which do not work on one request at a time (such as processor sharing policies) and which switch between requests in service at times other than arrival or departure epochs. Recall that these sets of policies were excluded by proof from the original system; here we exclude them *a priori*.

Given the above assumptions, we can derive an optimal service ordering policy from the set of policies in $\mathcal{S}'$ that satisfies the following requirements:

Lemma 5 *An optimal policy can be found among the class of non-idling policies.*

Proof: The proof is similar to that of Lemma 1. We construct the alternate policy p' by taking an infinitesimally small time interval of size dl from $[a_2, Z_1)$ and swapping it with an interval of size dl in $[a_1, a_2)$, the period over which the server idles while requests are queued. At most one departure can occur in

this interval with probability $u_k + o(dl)$, where k is the request that the server works on during the interval dl. Performing this switch increases the expected revenue by a nontrivial amount, establishing a contradiction. □

Lemma 6 *An optimal policy can be found among the class of Markov policies.*

Proof: The proof is similar to the proof of Lemma 4. The service time distributions of each request are exponential; therefore, service decisions will depend only on the present state of the system. □

Because we now have additional information about each request in terms of the expected service times, the state vector must incorporate this additional information:

Definition 3 *The* state vector *of this system at time t under a service policy p is*

$$G_{t_p} = \{(g_{1_p}(t), \mu_1) \ \ (g_{2_p}(t), \mu_2) \ \ \ldots \ (g_{M_p}(t), \mu_M)\}$$

where M and $g_{i_p}(t)$ are as defined in Definition 1 and $1/\mu_i$ is the expected remaining service time of request i.

The "best job" is now $\arg\max_i g_{i_p}(t)\,\mu_i$, $i \in 1, 2, \ldots, M$. That is, the best job corresponds to the request with the highest expected revenue weighted by the request's expected completion time.

Using an argument similar to the one presented in the proof of Theorem 1, we show that the following theorem holds:

Theorem 3 *The optimal policy for this system is greedy. At any time t, it chooses to serve the pending request with the highest $g_{i_p}(t)\,\mu_i$ product.*

Proof: We sketch the key idea of the proof here. We consider an optimal policy p which does the following: at time t, serve request k to completion, and upon k's departure, serve request $(k+1)$ to completion, where $g_{k_p}(t)\mu_k < g_{(k+1)_p}(t)\mu_{(k+1)}$. As in the proof of Theorem 1, we assume no arrivals from t until the end of the current busy cycle. We then construct an optimal policy p' which serves request $(k+1)$ to completion starting at time t and then serves request k to completion. Here, the proof deviates slightly from that of Theorem 1, because we cannot make the claim that the remaining service times of the two requests are distributed identically. Therefore, we take the difference in expected values of the revenue generated by policies p and p' from t until the end of the current busy cycle, which yields

$$\begin{aligned} E\left[\sum_{j=1}^{M} g_{j_{p'}}(t_{j_{p'}})\right] - E\left[\sum_{j=1}^{M} g_{j_p}(t_{j_p})\right] = \\ g_{(k+1)_p}(t)\frac{\mu_{(k+1)}}{c+\mu_{(k+1)}}\left(1 - \frac{\mu_k}{c+\mu_k}\right) \\ - g_{k_p}(t)\frac{\mu_k}{c+\mu_k}\left(1 - \frac{\mu_{(k+1)}}{c+\mu_{(k+1)}}\right) \\ = \quad \frac{c(g_{(k+1)_p}(t)\mu_{(k+1)} - g_{k_p}(t)\mu_k)}{(c+\mu_k)(c+\mu_{(k+1)})} \end{aligned} \tag{4}$$

where M is the number of requests in the system at time t. Clearly the numerator of (4) is positive, and therefore p is not optimal. □

The optimal policy for this system, then, chooses to serve the request with the highest $g_{i_p}(t)\,\mu_i$ product at any time.

4.2.1 A counterexample

Earlier in this section, we mentioned several restrictions on the set of possible optimal policies which were assumed rather than demonstrated by proof. We explain the reasoning behind these restrictions now in more detail by means of a counterexample.

Suppose there exists a sample path in which the server processes two requests during a busy cycle. We label these requests "job 1" and "job 2". There are no further arrivals to this system for the remainder of the current busy cycle. In this sample path, job 1 arrives at time zero, and job 2 arrives at some time $a > 0$, where a is less than the service time required by job 1. The busy cycle ends at some time $b, b > a$.

We consider three service policies, labeled NPS1, NPS2, and PS. NPS1 and NPS2 are selected from $\mathcal{S}'$, while PS is selected from $\mathcal{S}$ such that PS $\notin \mathcal{S}'$. The three policies behave as follows over the interval $[a, b]$:

- **NPS1**: The server processes job 1 to completion, queueing job 2. Upon job 1's departure, the server processes job 2 to completion.
- **NPS2**: The server processes job 2 to completion, queueing job 1. Upon job 2's departure, the server resumes processing job 1 to completion.
- **PS**: The server processes both job 1 and job 2 simultaneously, by means of some sort of resource sharing, until one of the jobs completes service and departs, at which time the server dedicates all of its resources to processing the remaining job to completion.

We are interested in the revenue the server expects to earn over this sample path under each of the three policies. We derive the general expressions for the expected revenue earned under the three policies first, and then describe a specific example where the expected revenue earned by the PS policy is greater than the expected revenues earned by the NPS1 and NPS2 policies separately.

The associated revenue decay for the two jobs at time a is $c_1 = e^{-ca}$ and $c_2 = e^{-c0} = 1$, respectively. We define T_1' as the remaining service time for job 1 beyond time a and T_2 as the service time for job 2 beyond time a. Due to the memoryless property of the exponential distribution, T_1 and T_1' are exponentially distributed with parameter μ_1; T_2 is exponentially distributed with parameter μ_2.

We consider the two non-processor sharing policies first. Under policy NPS1, the time job 1 spends in the system past time a is T_1'. Job 2's total time in the system is $T_1' + T_2$. The revenue earned by the server during this sample path under this policy

is $g_{\text{NPS1}} = c_1\, e^{-cT_1'} + c_2\, e^{-c(T_1'+T_2)}$. The expected revenue is given by the equation

$$E[g_{\text{NPS1}}] = \int_0^\infty \int_0^\infty [c_1 e^{-ct_1'} + c_2 e^{-c(t_1'+t_2)}] f_{T_1',T_2}(t_1', t_2) dt_1'\, dt_2$$

Because the service times are independent, this equation evaluates to

$$E[g_{\text{NPS1}}] = c_1 \frac{\mu_1}{c+\mu_1} + c_2 \frac{\mu_1}{c+\mu_1}\frac{\mu_2}{c+\mu_2} \quad (5)$$

In a similar manner, we derive the expected revenue earned by the server in policy NPS2:

$$E[g_{\text{NPS2}}] = c_1 \frac{\mu_1}{c+\mu_1}\frac{\mu_2}{c+\mu_2} + c_2 \frac{\mu_2}{c+\mu_2} \quad (6)$$

Under the PS policy, the server devotes a fraction q of its total resources to processing job 1 and $1-q$ of its total resources to processing job 2 while both requests are in the system, where $0 < q < 1$. Define T_f as the time at which the first job to complete service departs the system with respect to time a, where $T_f = \min(T_1', T_2)$ and is exponentially distributed with parameter $(q\mu_1 + (1-q)\mu_2) \equiv \mu$. Also, define T_s as the time at which the remaining job departs the system. We set $T_s = T_f + Z$, where Z is the remaining processing time for the job remaining in the system past time $T_f + a$. The probability that job 1 completes its service and departs the system first is $(q\mu_1)/\mu$, and the probability that job 2 completes its service and departs the system first is $[(1-q)\mu_2]/\mu$.

To define Z, we let $Z = Z_1$ given that job 1 finishes first, and $Z = Z_2$ given that job 2 finishes first. Thus, Z_1 is distributed exponentially with parameter μ_2 with probability $q\mu_1/\mu$ and Z_2 is distributed exponentially with parameter μ_1 with probability $(1-q)\mu_2/\mu$. The distribution of Z is given by the equation

$$\begin{aligned} f_Z(z) &= \frac{q\mu_1}{\mu} f_{Z_1}(z) + \frac{(1-q)\mu_2}{\mu} f_{Z_2}(z) \\ &= \frac{\mu_1\mu_2}{\mu}[q e^{-\mu_2 z} + (1-q) e^{-\mu_1 z}] \end{aligned}$$

We find that T_f and Z are independent; thus, the joint distribution of the service times of the two jobs is given by $f_{T_f}(t_f) f_Z(z) = \mu_1\mu_2 e^{-\mu t_f}[q e^{-\mu_2 z} + (1-q) e^{-\mu_1 z}]$. In addition, if $T_f = T_1'$, then we define $c_f \equiv c_1$ and $c_s \equiv c_2$; otherwise, we define $c_f \equiv c_2$ and $c_s \equiv c_2$.

The revenue collected by the server during the sample path is $g_{\text{PS}} = c_f e^{-cT_f} + c_s e^{-cT_s} = c_f e^{-cT_f} + c_s e^{-c(T_f+Z)}$, and the expected value of this revenue is $E[g_{\text{PS}}] = c_f\mu/(c+\mu) + c_s\mu_1/(c+\mu_1)\mu_2/(c+\mu_2)$.

If job 1 completes first with probability $(q\mu_1)/\mu$, then we make the proper substitutions and obtain

$$E[g_{\text{PS}}] = \frac{qc_1\mu_1 + (1-q)c_2\mu_2}{c+\mu} + \frac{1}{\mu}\frac{\mu_1}{c+\mu_1}\frac{\mu_2}{c+\mu_2}[q\, c_2\mu_1 + (1-q)c_1\mu_2] \quad (7)$$

We now show that there exist nonnegative values of $c_1, c_2, \mu_1, \mu_2, c$, and q for which the expected revenue for the PS policy exceeds the expected revenues of both policies NPS1 and NPS2. Let us assume that the server splits its resources evenly between the two jobs, such that $q = 0.5$. Let us also assume that $c = 1$. We have already established that $c_2 = 1$. We set $c_1 = 0.45$, $\mu_1 = 20$, and $\mu_2 = 10$. Plugging these values in to the revenue expressions yields $E[g_{\text{NPS1}}] = 1.2987$, $E[g_{\text{NPS2}}] = 1.2944$ and $E[g_{\text{PS}}] = 1.3008$. Clearly, $E[g_{\text{PS}}] > E[g_{\text{NPS1}}]$ and $E[g_{\text{PS}}] > E[g_{\text{NPS2}}]$. Thus, for this sample path, neither of the NPS revenues is strictly greater than the revenue generated by the PS policy, and the conjecture does not hold.

Thus, there exists at least one sample path in which a PS policy outperforms a pair of equivalent NPS policies. In fact, more sample paths like this one exist, and therefore we cannot completely eliminate PS policies from consideration in systems where file size distributions are variable.

5 Conclusions

We have shown that when network delays are ignored, an impatient user population is best served using a nontraditional, "greedy" service ordering policy. Server performance is maximized when the concept of fairness is ignored. The possible loss of revenue from requests that give up is compensated by the greater payoff from the requests that are served to completion.

Additionally, we presented a numerical example in which a processor sharing policy performs better than its non-processor-sharing counterparts. The implications of this are still under study, as we have not been able to reproduce this behavior under large sample paths in simulation.

References

[1] Nina Bhatti and Rich Friedrich. Web server support for tiered services. *IEEE Network*, 13(5):64–71, September/October 1999.

[2] Mark E. Crovella, Robert Frangioso, and Mor Harchol-Balter. "Connection scheduling in web servers". In *USENIX Symposium on Internet Technologies and Systems*, pages 243–254, Boulder, Colorado, October 1999.

[3] Amy Csizmar Dalal. *Characterization of User and Server Behavior in Web-Based Networks*. PhD thesis, Northwestern University, December 1999.

[4] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk Nielsen, and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, June 1999. RFC 2616.

[5] Mor Harchol-Balter, Mark E. Crovella, and Sung Sim Park. "The case for SRPT scheduling in web servers". Technical Report MIT-LCS-TR-767, MIT Laboratory for Computer Science, October 1998.

[6] Louis P. Slothouber. "A model of web server performance". StarNine Technologies, Inc.