A Constrained Optimization Problem for a Processor Sharing Queue

Peter de Waal
CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In this article we discuss a processor sharing queueing model for a stored program controlled telephone exchange. The model incorporates the effects of both call requests and operator tasks on the load of the processor. Newly arriving call requests and operator tasks can either be admitted or rejected and for this decision the state of the queue at the moment of arrival is available as information. To guarantee a high level of service, we formulate a constrained optimization problem. Two types of access control, viz., partitioning policies and sharing policies, are considered. The optimization problem is solved for partitioning policies and the performance of both types of policies is compared. © 1993 John Wiley & Sons, Inc.

1. INTRODUCTION

In this article we discuss a processor sharing (PS) queueing model for a *stored program controlled* (*SPC*) telephone switch. We consider two different customer types who arrive to the queue, representing call requests and operator tasks. Newly arriving customers may be admitted to the queue or rejected, depending on the state of the queue. The control objective is to maximize the throughput of call requests subject to two constraints: an upper bound on the call requests' mean sojourn time and a lower bound on the fraction of admitted operator tasks.

This PS queue is a model for the processor in a stored program controlled telephone exchange. Maximization of revenue corresponds to maximization of the number of completed call requests. The bound on the mean sojourn time of these requests guarantees that the fraction of calls that are lost due to impatience will be small. The bound on the throughput of operator tasks guarantees that a minimum fraction of processing power is always available for, e.g., testing and administrative procedures.

In this article we shall restrict our attention to a special class of admission policies, viz., *partitioning policies*. In a partitioning policy the decision whether to admit a new request or task depends on two separate thresholds, one for each customer class. If the number of call requests in service reaches the corresponding threshold, new requests are rejected until the number of requests in service decreases by a service completion. An analogous algorithm is employed for operator tasks. This type of policy is called a partitioning policy, because its effect is basically a partitioning of the processor power over the two customer classes. We shall also briefly discuss a second type of admission policies, viz.,

Naval Research Logistics, Vol. 40, pp. 719–731 (1993)

CCC 0894-069X/93/050719-13

sharing policies. This type of control was implemented to allow "sharing" of the processor capacity, i.e., when the demand for service of call requests is low, then operator tasks are allowed to use part of the call-handling capacity and vice versa. The terms partitioning and sharing were used in Kaufman [4] in the context of memory allocation policies for computers.

In this article we shall discuss the constrained control problem, that was introduced in the beginning of this section, for both classes of admission policies. We prove that the problem can be solved for partitioning policies. Although we have not been able to solve the problem for sharing policies, we shall briefly discuss the performance of such policies.

Constrained optimization problems for queues have gained increasing interest of researchers during the last five years. In Ross [12] finite-state Markov decision processes are considered with a reward and cost structure. The objective is to maximize the average return with a constraint on the average costs. In Hordijk and Spieksma [3] an optimization problem with a constraint on the average cost for a one-dimensional queueing system is discussed. They show that for rather general assumptions on the cost and reward structure the optimal control randomizes in exactly one state. In Ma and Makowski [7] a constrained optimal flow control problem is solved with Lagrangian methods. Again the optimal control is shown to be randomized in exactly one state. A well-known reference on constrained optimal flow control is Robertazzi and Lazar [11]. They consider the problem of maximizing the throughput of a one-dimensional queue under a constraint on the average delay. The optimal control is shown to be of a window flow type and again it is randomized in exactly one state. In Nain and Ross [8] a queueing model is discussed for multiplexing heterogeneous Poisson arrival streams onto a single communication channel. The optimization problem is to minimize a linear combination of the average delays, while at the same time subjecting the average delay for one stream to a hard constraint. The optimal multiplexing policy is shown to be a randomized modification of the μc rule. In Nain and Ross [9] a similar system is considered with renewal arrival streams. To minimize a linear combination of the average queue lengths with a hard constraint on the average queue length of one arrival stream, the optimal multiplexing policy is shown to be a randomized modification of a static-priority rule. The performance analysis of partitioning and sharing policies is related to the analysis of resource sharing in Kaufman [4]. He derives a product form for the equilibrium distribution of a multiserver queueing model with sharing policies and presents a one-dimensional recursion for the computation of performance measures. In Foschini and Gopinath [2] an optimal control problem for a queueing system with two processors and a common waiting room is presented. They show that the control that minimizes a weighted sum of the idle times of the servers is a combination of a partitioning and a sharing policy.

This article is organized as follows. In Section 2 we present a description of the queueing model and formulate the control problem. In Section 3 we derive monotonicity properties of performance measures that are used in Section 4 to establish conditions for the existence of an optimal policy. A design algorithm for the optimal policy is discussed in Section 5. We briefly address the performance of sharing policies in Section 6. The article is concluded with some numerical examples in Section 7.

2. DESCRIPTION OF THE QUEUEING MODEL

Consider a PS queue with two independent Poisson arrival processes. Customers of class 1 represent the call requests and they arrive with rate λ_1. Operator tasks are referred to as class 2 customers and they arrive with rate λ_2. The service requirements for customers of class i are independent and may have some general type distribution function with mean $1/\mu_i$, $i = 1, 2$. We define the workloads $\rho_i = \lambda_i/\mu_i$, $i = 1, 2$.

The queueing process is a continuous-time Markov process on the state space $\mathcal{S} = \mathbb{N}^2$, where a state is represented by a population vector $\mathbf{m} = (m_1, m_2)$ corresponding to a population of m_i customers of class i, $i = 1, 2$. Upon arrival to the queue customers can either be admitted or rejected according to a stationary admission policy. A partitioning policy is described by two thresholds, $M_1, M_2 \in \mathbb{N} \cup \infty$. A new customer of type i, $i = 1, 2$, is admitted only if the number of customers of type i at the moment of arrival is smaller than M_i. The effect of applying such an admission policy is that the state space is restricted to the set $\{(m_1, m_2) | 0 \leq m_i \leq M_i, i = 1, 2\}$. We shall use the notation $\langle M_1, M_2 \rangle$ to denote both this subset of $\mathbb{N}^2$ and the partitioning policy itself.

We let $T_i(M_1, M_2)$ denote the throughput of type i, $i = 1, 2$, for policy $\langle M_1, M_2 \rangle$. Similarly we let $S_i(M_i, M_2)$ denote the mean sojourn time of type i customers, as a function of the thresholds. In case either or both of the thresholds are infinite, then we need to impose the usual conditions on the workload to ensure the ergodicity of the queue:

- If $M_1 = \infty$, then $\rho_1 < 1$.
- If $M_2 = \infty$, then $\rho_2 < 1$.
- If $M_1 = M_2 = \infty$, then $\rho_1 + \rho_2 < 1$.
- If both M_1 and M_2 are finite, then ρ_1 and ρ_2 may take any value.

With these definitions and conditions we are now all set to introduce the constrained control problem.

DEFINITION 2.1 (Problem (P)): Let S and T be fixed constants in $\mathbb{R}_+$ satisfying $S \geq 1/\mu_1$ and $T \leq \lambda_2$.

1. A partition $\langle M_1, M_2 \rangle$ is called *feasible* if $S_1(M_1, M_2) \leq S$ and $T_2(M_1, M_2) \geq T$.
2. A partition $\langle M_1, M_2 \rangle$ is called *optimal* if it is feasible and if there exists no other feasible policy $\langle M_1^*, M_2^* \rangle$ with $T_1(M_1^*, M_2^*) > T_1(M_1, M_2)$.

The condition $S \geq 1/\mu_1$ is to guarantee that the constrained problems are not trivial, since the mean sojourn time of a class 1 customer can never be smaller than its mean service time. For the same reason the condition $T \leq \lambda_2$ was introduced, since the throughput of class 2 customers can never be larger than the arrival rate.

Due to the special type of blocking for newly arriving customers the equilibrium joint queue length distribution has a product form.

LEMMA 2.2: The equilibrium joint queue length distribution $p(n_1, n_2)$ of having n_1 and n_2 customers of type 1 and 2, when the partitioning policy is

$\langle M_1, M_2 \rangle$, is given by

$$p(n_1, n_2) = \begin{cases} \dfrac{1}{G(M_1, M_2)} \binom{n_1+n_2}{n_1} \rho_1^{n_1} \rho_2^{n_2}, & \text{if } (n_1, n_2) \in \langle M_1, M_2 \rangle, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$G(M_1, M_2) = \sum_{(n_1,n_2)\in\langle M_1,M_2\rangle} \binom{n_1+n_2}{n_1} \rho_1^{n_1} \rho_2^{n_2}$$

is a normalizing constant.

PROOF: See, for instance, Cohen [1] or Kelly [5, Corollary 1.10].

For finite thresholds the performance measure that we consider (throughputs and mean sojourn times) can be computed quite effectively by a recursive scheme in M_1 and M_2 (cf. Reiser and Kobayashi [10]). For infinite thresholds we can also derive expressions of $T_i(M_1, M_2)$ and $S_i(M_1, M_2)$, that can be computed without considerable effort (see de Waal [15, Chap. 5]).

3. MONOTONICITY PROPERTIES

In this section we shall discuss monotonicity results for these performance measures with respect to the thresholds. These results are employed to formulate existence conditions for an optimal policy in Section 4 and to design a search procedure for the optimal thresholds in Section 5.

LEMMA 3.1 (monotonicity properties): For all $M_1, M_2 \in \mathbb{N}$ and $i = 1, 2$,

$$T_1(M_1, M_2) < T_1(M_1 + 1, M_2), \qquad T_1(M_1, M_2) > T_1(M_1, M_2 + 1), \quad (1)$$

$$T_2(M_1, M_2) > T_2(M_1 + 1, M_2), \qquad T_2(M_1, M_2) < T_2(M_1, M_2 + 1), \quad (2)$$

$$L_i(M_1, M_2) < L_i(M_1 + 1, M_2), \qquad L_i(M_1, M_2) < L_i(M_1, M_2 + 1), \quad (3)$$

$$S_i(M_1, M_2) < S_i(M_1 + 1, M_2), \qquad S_i(M_1, M_2) < S_i(M_1, M_2 + 1). \quad (4)$$

PROOF: See de Waal and van Dijk [16].

Note that these inequalities are indeed what we would intuitively expect. For example, (1) states that increasing the threshold M_1 for class 1 while keeping M_2 constant gives a higher throughput T_1, since obviously more class 1 customers are going to be admitted to the queue. Conversely the throughput T_2 for class 2 will decrease as less processor capacity will be left to serve class 2 customers.

In de Waal and van Dijk [16] the monotonicity properties of Lemma 3.1 are proven for a service discipline that is more general than standard PS. This

generalization includes the model in Cohen [1], and also nonsymmetric modifications. The monotonicity of throughputs and mean queue lengths is proven by treating these performance measures as average rewards and showing that monotonicity exists for finite horizon rewards. The inequality (4) for S_2 is an immediate result of (2) and (3) by application of Little's law. The inequality (4) for S_1 cannot be derived in this manner and has to be proven explicitly. This is the only monotonicity property where the product form equilibrium distribution is actually needed.

The monotonicity results of Lemma 3.1 remain valid when one of the thresholds becomes infinite, though for obvious reasons the strict inequality signs $<$ and $>$ have to be replaced by $\leq$ and $\geq$, respectively.

4. EXISTENCE OF AN OPTIMAL PARTITION

We shall now use the monotonicity results of the preceding section to give conditions for the existence of a solution to problem (P). The first lemma shows that the existence of a feasible partition is sufficient for the existence of an optimal partition. This result is not trivial, since the existence of a feasible partition does not guarantee that there is a partition that attains the maximum class 1 throughput.

LEMMA 4.1: There exists a feasible policy for problem (P), if and only if there exists an optimal policy.

PROOF: Since an optimal partition is by definition feasible, the "only if" statement is immediate. The proof for the "if" part proceeds by contradiction. Assume that there is a feasible partition, but no optimal partition. Since T_1 is bounded by λ_1, this means that there is no feasible partition that attains a maximum class 1 throughput and thus there must be infinitely many feasible partitions. Consequently there exists a sequence $\{\mathscr{C}_n\}_{n=0}^{\infty}$ of feasible partitions satisfying

(a.i) $T_1(\mathscr{C}_n)$ is strictly increasing for $n \in \mathrm{N}$.
(a.ii) For any feasible partition $\mathscr{C}$ there exists an $n_{\mathscr{C}} \in \mathrm{N}$ such that $T_1(\mathscr{C}_n) > T_1(\mathscr{C})$ for $n > n_{\mathscr{C}}$.

In the remainder of this proof we shall use the notation $M_i(\mathscr{C})$ for $i = 1, 2$, to denote the class i threshold of partition $\mathscr{C}$. Note that $M_i(\mathscr{C})$ can be infinite.

First we shall prove that

$$\sup_{n \in \mathrm{N}} M_1(\mathscr{C}) = \infty. \tag{5}$$

Assume that (5) is not true, so $M_1(\mathscr{C}_n)$ takes only a finite number of discrete values. Consequently there must exist a subsequence $\{\mathscr{C}_n^{(1)}\}_{n=0}^{\infty}$ of $\{\mathscr{C}_n\}_{n=0}^{\infty}$, with $M_1(\mathscr{C}_n^{(1)})$ constant and equal to $\limsup_{n\to\infty} M_1(\mathscr{C}_n)$ which we shall denote as $\overline{M}_1$. Since $T_1(\mathscr{C}_n)$ and thus also $T_1(\mathscr{C}_n^{(1)})$ is strictly increasing, $M_2(\mathscr{C}_n^{(1)})$ must

take infinitely many different values due to (1). Therefore, we can construct a subsequence $\{\mathscr{C}_n^{(2)}\}_{n=0}^{\infty}$ of $\{\mathscr{C}_n^{(1)}\}_{n=0}^{\infty}$ such that

(b.i) $M_1(\mathscr{C}_n^{(2)})$ is constant (by definition of $\mathscr{C}_n^{(1)}$ and thus also of $\mathscr{C}_n^{(2)}$).
(b.ii) $M_2(\mathscr{C}_n^{(2)})$ is strictly increasing in $n \in \mathrm{N}$.
(b.iii) $T_1(\mathscr{C}_n^{(2)})$ is strictly increasing in $n \in \mathrm{N}$.

This is in contradiction with the monotonicity property (1) of Lemma 3.1, however. We may thus conclude that $\sup_{n\in\mathrm{N}} M_1(\mathscr{C}_n) = \infty$.

If we have a feasible partition $\mathscr{C}_n$ for some $n \in \mathrm{N}$ with $M_1(\mathscr{C}_n) = \infty$, then $T_1(\mathscr{C}_n) = \lambda_1$ (no type 1 customers are rejected), and this contradicts (a.ii). We may therefore assume that for all $n \in \mathrm{N}$, $M_1(\mathscr{C}_n) < \infty$. Since $\sup_{n\in\mathrm{N}} M_1(\mathscr{C}_n) = \infty$, there must exist a subsequence $\{\mathscr{C}_n^{(3)}\}_{n=0}^{\infty}$ of $\{\mathscr{C}_n\}_{n=0}^{\infty}$ with $M_1(\mathscr{C}_n^{(3)}) \uparrow \infty$ as $n \to \infty$. With regard to the sequence $\{M_2(\mathscr{C}_n^{(3)})\}_{n=0}^{\infty}$ we can distinguish between the following two cases:

(c.i) $M_2(\mathscr{C}_n^{(3)})$ takes only finitely many values for $n \in \mathrm{N}$.
(c.ii) $M_2(\mathscr{C}_n^{(3)})$ takes infinitely many values for $n \in \mathrm{N}$.

If (c.i) holds, then we can construct (yet another) subsequence $\{\mathscr{C}_n^{(4)}\}_{n=0}^{\infty}$ of $\{\mathscr{C}_n^{(3)}\}_{n=0}^{\infty}$, such that $M_2(\mathscr{C}_n^{(4)})$ is constant and equal to $\operatorname{limsup}_{n\to\infty} M_2(\mathscr{C}_n^{(3)})$ which we shall denote as $\overline{M}_2$. For all $n \in \mathrm{R}$ we then have by the definition of feasibility, $S_1(M_1(\mathscr{C}_n^{(4)}), \overline{M}_2)$ increasing and bounded from above by S, and $T_2(M_1(\mathscr{C}_n^{(4)}), \overline{M}_2)$ decreasing and bounded from below by T. Furthermore, by definition of $\{\mathscr{C}_n^{(3)}\}_{n=0}^{\infty}$, we have $M_1(\mathscr{C}_n^{(4)}) \uparrow \infty$ as $n \to \infty$ and thus $\langle \infty, \overline{M}_2 \rangle$ is a feasible partition. Since $T_1(\infty, \overline{M}_2) = \lambda_1$, this leads to a contradiction with (a.ii).

Finally in case (c.ii) it is possible to construct a subsequence $\{\mathscr{C}_n^{(5)}\}_{n=0}^{\infty}$ of $\{\mathscr{C}_n^{(3)}\}_{n=0}^{\infty}$ with $M_2(\mathscr{C}_n^{(5)})$ strictly increasing to ∞ for $n \in \mathrm{N}$. Since by construction $M_1(\mathscr{C}_n^{(3)})$ and thus also $M_1(\mathscr{C}_n^{(5)})$ is strictly increasing, we may conclude that $\langle \infty, \infty \rangle$ is a feasible partition. Since $T_1(\infty, \infty) = \lambda_1$, we get again a contradiction with (a.ii). As a result we may thus conclude that there exists an optimal partition. □

With this lemma we can now proceed with sufficient conditions for the existence of feasible partitions. The first sufficient condition for existence concerns a queueing model where the parameters ρ_1 and ρ_2 and the bound S are chosen such that the constraint of S_1 is satisfied by all partitioning policies.

LEMMA 4.2: If the queue is ergodic for the partitioning policy with thresholds $M_1 = M_2 = \infty$, i.e., $\rho_1 + \rho_2 < 1$, and if

$$S \geq \frac{1}{\mu_1(1 - \rho_1 - \rho_2)} \tag{6}$$

then $M_1 = M_2 = \infty$ is the optimal partitioning policy.

PROOF: Note that if $\rho_1 + \rho_2 < 1$, then the right-hand side of (6) is the expression for $S_1(\infty, \infty)$ (cf. Reiser and Kobayashi [10, Eq. (37)]). If the inequality (6) holds, then we know from Lemma 3.1 that $S_1(M_1, M_2) \leq S$ for all M_1, $M_2 \in \mathrm{N}$. Furthermore, $T_1(M_i, M_2) \leq \lambda_i$ for $M_1, M_2 \in \mathrm{N}$ and $i = 1, 2$, and this

bound is tight for $M_1 = M_2 = \infty$. Therefore $\langle \infty, \infty \rangle$ is a feasible policy, since $T_2(\infty, \infty) \geq T$. The class 1 throughput $T_1(\infty, \infty) = \lambda_1$, so $\langle \infty, \infty \rangle$ is the optimal partition. □

The following two lemmas give necessary and sufficient conditions for less trivial parameter settings.

LEMMA 4.3: If the lower bound $T = \lambda_2$, then there exists an optimal partition if and only if $S_1(1, \infty) \leq S$.

PROOF: Observe that the throughput $T_2(\mathscr{C})$ of class 2 under policy $\mathscr{C}$ is equal to the arrival rate λ_2 times the probability that a class 2 customer is admitted under $\mathscr{C}$. Since the blocking probability for class 2 is zero only for partitions with threshold $M_2 = \infty$, a feasible policy must thus use an infinite class 2 threshold. If $S_1(1, \infty) > S$, then by the inequality (4) we have $S_1(M_1, \infty) > S$ for all $M_1 \in \mathbb{N}$ and thus no feasible partition exists. If $S_1(1, \infty) \leq S$, then there exists at least one feasible partition, viz., $\langle 1, \infty \rangle$, and thus there exists an optimal partition. □

LEMMA 4.4: If $T < \lambda_2$, then there exists an optimal partition if and only if $S \geq S_1(1, K)$, where K is defined as

$$K = \min\{M_2 \in \mathbb{N} | T_2(1, M_2) \geq T\}. \tag{7}$$

PROOF: Let $T < \lambda_2$ and let K be defined as in (7). Such a K exists and is finite since $T_2(1, M_2) \uparrow \lambda_2$ as $M_2 \to \infty$. From the monotonicity of T_2 we know that a partition $\langle M_1, M_2 \rangle$ can be feasible only if M_2 is at least K. If $S_1(1, K) \leq S$, then $\langle 1, K \rangle$ is a feasible partition and thus an optimal policy exists. If $S_1(1, K) > S$, then by (4) for all $M_1 \geq 1$ and $M_2 \geq K$ we have $S_1(M_1, M_2) > S$, and consequently there are no feasible policies. □

We can derive closed-form expressions for $S_1(1, M_2)$ and $T_2(1, M_2)$, $M_2 \in \mathbb{N}$, that can be computed recursively in M_2. The monotonicity of S_1 and T_2 can be employed to search for a K satisfying (7).

5. DESIGN OF THE OPTIMAL PARTITION

Once the existence of a feasible partition has been established, we can use the monotonicity properties of Lemma 3.1 to design a procedure to find the optimal partition. Assume for instance that we have a partition $\langle M_1, M_2 \rangle$ that violates the constraint $S_1(M_1, M_2) \leq S$. From the monotonicity of S_1 we then know that a feasible partition $\langle M_1^*, M_2^* \rangle$ must satisfy $M_i^* \leq M_i$, $i = 1, 2$.

Since we have expressions for all the performance measures that are involved in the formulation of problem (P), we are all set to discuss the algorithm to find the optimal partition. In this procedure we can distinguish the following steps.

(i) Test if there exists a feasible partition.
(ii) Determine the set of feasible partitions.
(iii) Restrict the set of feasible partitions to a finite set that contains the optimal partition.
(iv) Find the optimal partition over this finite set.

It is not necessary in all cases to perform all of these four steps, since for some examples we can already conclude in steps (i) or (ii) what the optimal partition is (if any).

For step (i) we use the results that were obtained in Lemmas 4.2–4.4. When we have a situation where Lemma 4.2 applies, then obviously we can stop, since we have found an optimal partition. If this is not the case, then we can check whether we have an example of Lemma 4.3. If this is true, then we know from Lemma 4.3 that the optimal partition must be of the form $\langle M_1, \infty \rangle$ for some $M_1 \in \mathbb{N}$. The value of $S_1(M_1, \infty)$ can be computed quite easily (see de Waal [15, Chap. 5]). If there exists a feasible partition, then clearly the optimal partition is $\langle M_1^*, \infty \rangle$, with

$$M_1^* = \max\{M_1 \in \mathbb{N} | S_1(M_1, \infty) \leq S\}.$$

Such an M_1^* exists and is finite, since $S_1(M_1, \infty) \uparrow \infty$ as $M_1 \to \infty$ (otherwise the conditions of Lemma 4.2 would have been satisfied).

If we can exclude the cases where Lemma 4.2 and 4.3 apply, then we are in a situation where at least one of the two thresholds of the optimal partition must be finite. We can then characterize the set of feasible policies. We do this by determining for any fixed value of M_2 the maximally feasible class 1 threshold. We define for all $M_2 \in \mathbb{N} \cup \{\infty\}$:

$$\mathcal{M}_1^T(M_2) = \sup\{M_1 \in \mathbb{N} | T_2(M_1, M_2) \geq T\},$$

$$\mathcal{M}_1^S(M_2) = \sup\{M_1 \in \mathbb{N} | S_1(M_1, M_2) \leq S\},$$

$$\mathcal{M}_1(M_2) = \min\{\mathcal{M}_1^T(M_2), \mathcal{M}_1^S(M_2)\}.$$

For any fixed M_2 a partition $\langle M_1, M_2 \rangle$ can be feasible only if $M_1 \leq \mathcal{M}_1(M_2)$. An immediate result from the monotonicity properties is the local optimality of the class 1 threshold $\mathcal{M}_1(M_2)$ for fixed M_2, since for that threshold the maximal class 1 throughput is obtained.

The set of all feasible partitions can now be characterized as

$$\{\langle M_1, M_2 \rangle | M_2 \in \mathbb{N} \cup \{\infty\}, 0 \leq M_1 \leq \mathcal{M}_1(M_2)\},$$

and for optimization purposes this set can be restricted to

$$\{\langle \mathcal{M}_1(M_2), M_2 \rangle | M_2 \in \mathbb{N} \cup \{\infty\}\},$$

that contains the optimal partition. Examination of the structure of this set reveals that there are two difficulties in the evaluation of T_1 over this set. First we see that we have infinitely many values for M_2 and second we can encounter the situation where $\mathcal{M}_1(M_2) = \infty$ for some M_2. With respect to the latter possibility we can remark that this is in fact no difficulty, since if for some M_2 the partition $\langle \infty, M_2 \rangle$ is feasible, then it is also optimal since $T_1(\infty, M_2) = \lambda_1$.

The remaining problem thus concerns the infinitely many values that M_2 can take. We shall show that we can restrict ourselves to a finite set, thus reducing the search of the optimal partition to a search over a finite set. For this we need the following lemma.

LEMMA 5.1:

(i) $\mathcal{M}_1^T(m)$ is nondecreasing in m.
(ii) $\mathcal{M}_1^S(m)$ is nonincreasing in m.
(iii) $\mathcal{M}_1(m)$ is unimodal in m; i.e., $\mathcal{M}_1(m-1) \geq \mathcal{M}_1(m)$ implies $\mathcal{M}_1(m) \geq \mathcal{M}_1(m+1)$, and $\mathcal{M}_1(m) \leq \mathcal{M}_1(m+1)$ implies $\mathcal{M}_1(m-1) \leq \mathcal{M}_1(m)$.

PROOF: Parts (i) and (ii) are immediate from the monotonicity of T_2 and S_1, respectively. Part (iii) follows from the definition of $\mathcal{M}_1$ and (i) and (ii). □

Recall from the previous steps of the algorithm that we can exclude the possibility of a feasible partition with two infinite thresholds. We may therefore conclude by the unimodality of $\mathcal{M}_1$ that $\mathcal{M}_1(m)$ converges to a finite value $\mathcal{M}_1(\infty)$ as $m \to \infty$. This limit can be computed from the expressions for $S_1(M_1, \infty)$ and $T_2(M_1, \infty)$. Since $\mathcal{M}(m)$ can thus take only finitely many values in $\mathbb{N} \cup \{\infty\}$, there must be a finite $\mathcal{M}_2$ such that $\mathcal{M}_1(M_2) = \mathcal{M}_1(\infty)$ for $M_2 \geq \mathcal{M}_2$. From the monotonicity of T_1 it is clear that within the set

$$\{\langle \mathcal{M}_1(M_2), M_2\rangle | M_2 \geq \mathcal{M}_2\}$$

the maximum class 1 throughput is obtained for partition $\langle \mathcal{M}_1(\mathcal{M}_2), \mathcal{M}_2\rangle$. Therefore it is sufficient to restrict the search for the optimal partition to the finite set

$$\{\langle \mathcal{M}_1(M_2), M_2\rangle | 0 \leq M_2 \leq \mathcal{M}_2\}.$$

6. SHARING POLICIES

In this section we shall briefly discuss sharing policies. Sharing policies are characterized by a state space of the form

$$\{(m_1, m_2) \in \mathbb{N}^2 | 0 \leq c_1 m_1 + c_2 m_2 \leq c\},$$

where c_1 and c_2 are the *weight factors* for class 1 and 2, respectively, and c is called the *pool size*. The terms are derived from the actual implementation of these admission policies. It uses a pool that contains c permission units (p.u.'s) for some $c \in \mathbb{N}$. If a customer of type i, $i = 1, 2$, arrives at the queue, it can be admitted only if there are at least c_i p.u.'s left in the pool. If these p.u.'s are available, then they are removed from the pool and the customer is admitted; otherwise he is rejected. After an admitted customer completes his service, the p.u.'s for this customer are put back into the pool.

PS queues with sharing policies are related to the so-called stochastic knapsack problem. This model, presented in Ross and Tsang [13] and Ross and Yao [14], is similar to ours, but the PS queue is replaced by a multiserver station.

We know from Lam [6] and de Waal and van Dijk [17] that for sharing policies the equilibrium distribution is also given by a product form. This means that the performance measures can be evaluated quite easily. The main part of the algorithm is the computation of the normalization constant. Since we have only two customer types, this does not present any difficulty on modern workstations.

For example, an (unusually large) example with pool size 400 and both weight factors equal to 1 can be evaluated by complete enumeration within 15 seconds on a Sun Sparcstation 1, or within 3 seconds on an SGI Indigo 4000.

In contrast with the partitioning policies, there does not exist monotonicity of the performance measures with respect to the weight factors c_1, c_2, and the pool size c. This can be proven explicitly for some simple sharing policies (see de Waal [15, Chap. 5]). Unfortunately the monotonicity results for stochastic knapsack problems do not carry over to this PS queue, mainly because the processor sharing mechanism gives a stronger interaction between the two customer types than a multiserver discipline does.

7. NUMERICAL RESULTS

In this section we shall discuss the performance of partitioning and sharing policies from numerical examples. For several parameter settings of a queueing model the optimal partitions are computed and the robustness of this class of policies is discussed. For the same set of parameters we compute sharing policies of which the performance is close to the corresponding optimal partitioning policies. The robustness of sharing policies is also included in the exposition.

The queueing system we consider has class 2 arrival rate $\lambda_2 = 0.3$, and service rates $\mu_1 = \mu_2 = 1.0$. The lower bound on the class 2 throughput is set at $T = 0.2$ and the upper bound on the mean class 1 sojourn time is $S = 20$. In Table 1 the values of the optimal partitions' thresholds are presented for different values of the class 1 arrival rate λ_1. Included in the table are the values of the performance measures for these partitions. The third column in the table refers to the so-called *normalized arrival rate* $\bar{\lambda}_1$. Due to the lower bound $T = 0.2$ on the class 2 throughput, only a fraction 0.8 of the processor's speed is available to serve class 1 customers. Since $\mu_1 = 1.0$, this means that $\lambda_1 = 0.8$ corresponds to a call request load of 100%. For this reason the normalized arrival rate is introduced, and it is defined as $\bar{\lambda}_1 = \lambda_1/0.8$.

From Lemma 4.2 we know that for $\lambda_1 \leq 0.65$ ($\bar{\lambda}_1 \leq 0.81$) the mean sojourn time of class 1 will always be smaller than 20, and thus partition $\langle \infty, \infty \rangle$ is feasible and optimal. For $\lambda_1 \leq 0.80$ ($\bar{\lambda}_1 \leq 1.00$) the optimal class 1 threshold is infinite.

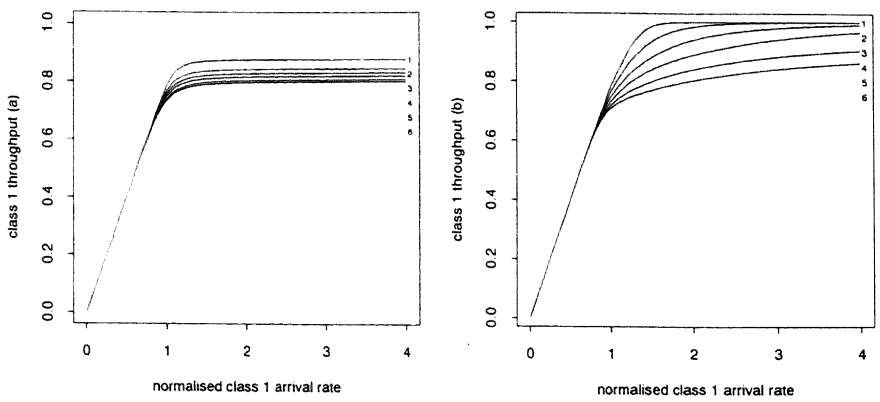

Figure 1. Throughput of class 1 for partitioning (a) and sharing (b) policies.

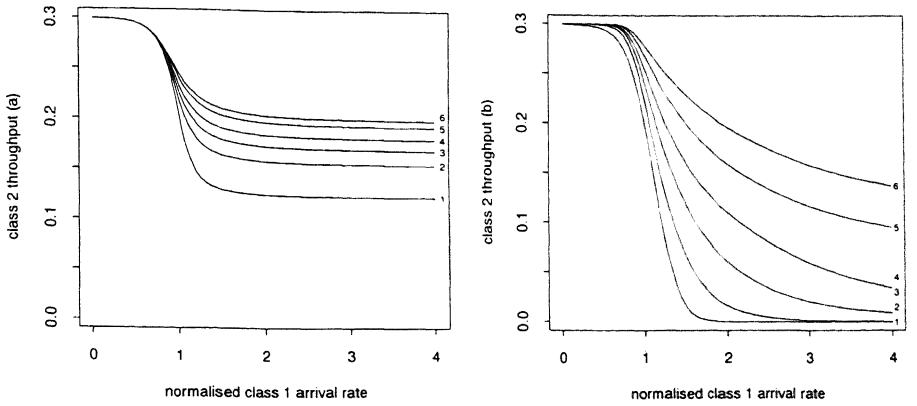

Figure 2. Throughput of class 2 for partitioning (a) and sharing (b) policies.

We have not included these policies in the results, since they are uninteresting when we come to the robustness issue. Because of the infinite theshold, the queue will no longer be ergodic when $\bar{\lambda}_1 \geq 1.00$.

In Figures 1(a)–3(a) the performance measures for the six partitions of Table 1 are depicted when the class 1 arrival rate is varied while the thresholds are kept constant. The numbers in the figures correspond to the numbers of the partitions in Table 1. From Figure 1(a) we can conclude that the largest values of T_1 can be obtained by using partition No. 1—optimal for $\bar{\lambda}_1 = 1.0$—for all arrival rates. This observation can be explained immediately from the monotonicity properties of Lemma 3.1. Due to the large thresholds of this partition the bounds on T_2 and S_1 are violated for large $\bar{\lambda}_1$, however. Conversely, partition No. 6 that is optimal for $\bar{\lambda}_1 = 1.5$ satisfies the constraints for all values of $\bar{\lambda}_1 \in [0.0, 4.0]$ but gives for $\bar{\lambda}_1 = 1.0$ a class 1 throughput that is about 7% less than the optimal value at that load. If one considers this loss to be too large, then clearly partitioning policies are not robust.

Next we discuss the performance of sharing policies. We have considered the PS queueing model for the same parameter settings as the partitioning policies.

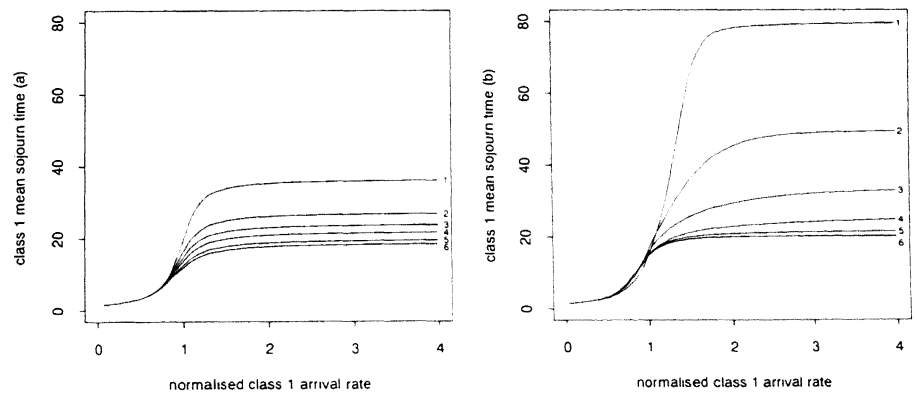

Figure 3. Mean sojourn time of class 1 for partitioning (a) and sharing (b) policies.

Table 1. Optimal partitioning policies.

No.	λ_1	$\bar{\lambda}_1$	M_1^{opt}	M_2^{opt}	T_1	T_2	S_1
1	0.80	1.00	32	5	0.788	0.201	19.7
2	0.85	1.06	23	5	0.792	0.202	18.8
3	0.90	1.13	20	5	0.794	0.203	18.2
4	1.00	1.25	18	5	0.798	0.201	18.2
5	1.20	1.50	16	5	0.797	0.203	17.7
6	1.50	1.88	15	5	0.796	0.204	17.5

From Section 6 we know that sharing policies do not exhibit monotonicity, that can be used to search for optimal policies. The performance measures, however, can be computed quite efficiently, thus allowing an "interactive" search for satisfying policies, i.e., policies of which the performance is close to the optimal partitioning policies. In the examples we found that the following rules can be applied in the search for a satisfying sharing policy if $\bar{\lambda}_1 \geq 1.0$. First we have to search for the right ratio of the weight factors to get the throughputs T_1 and T_2 close to 0.8 and 0.2, respectively. Having found the right ratio, we then have to vary the pool size c until S_1 is close to 20. In most cases we were able to find sharing policies of which the performance was close to that of partitioning policies (see Table 2). From this table we can conclude that when $\bar{\lambda}_1$ increases, then the ratio of c_1 and c_2 has to shift to give class 2 a preference over class 1 (compare policies No. 1 and No. 6).

Analogously to the investigation on partitioning policies we have examined the robustness of sharing policies. The results of these computations are shown in Figures 1(b)–3(b). From these figures we can draw the same conclusions as for the partitioning policies, viz., the policy that is optimal for $\bar{\lambda}_1 = 1.0$ violates the constraints when the arrival intensity increases. In this case the sharing policy that is optimal for $\bar{\lambda}_1 = 1.5$ is also not feasible for $\bar{\lambda}_1 = 4.0$. The loss of class 1 throughput at $\bar{\lambda}_1 = 1.00$ of sharing policy No. 6 compared to No. 1 is 9%, so sharing policies are not robust either. For some sharing policies the class 2 throughput even decreases to zero when the load becomes high. This is due to the triangular form of the state space. When the load of type 1 is high, the equilibrium distribution will have almost all of its mass in a state with a zero class 2 length.

For both classes of admission policies we conjecture that the performance under varying loads can be improved by an adaptive control scheme. There are two approaches to construct adaptive partitioning or sharing policies. In the first

Table 2. Satisfying sharing policies.

No.	λ_1	$\bar{\lambda}_1$	c_1	c_2	c	T_1	T_2	S_1
1	0.80	1.00	1	15	80	0.781	0.200	15.2
2	0.85	1.06	1	6	50	0.795	0.200	19.8
3	0.90	1.13	1	3	34	0.793	0.205	19.7
4	1.00	1.25	4	7	105	0.798	0.201	19.3
5	1.20	1.50	12	11	263	0.799	0.201	20.0
6	1.50	1.88	5	3	99	0.797	0.203	19.9

approach we determine the optimal or satisfying policies for a number of values of λ_1. In the exchange the value of this parameter is estimated periodically and the policy is adjusted according to the latest estimate. This type of control is called *certainty equivalence adaptive control.*

In the second approach we use one or several system characteristics as indicators for the value of λ_1. As an example one can think of the queue length, since a large number of customers may indicate that the arrival rate is high. In this approach the admission policy should be adjusted periodically according to the actual queue length.

REFERENCES

[1] Cohen, J.W., "The Multiple Phase Service Network with Generalized Processor Sharing," *Acta Informatica*, **12**, 245–284 (1979).

[2] Foschini, G.J., and Gopinath, B., "Sharing Memory Optimally," *IEEE Transactions on Communications*, **COM-31**, 352–359 (1983).

[3] Hordijk, A., and Spieksma, F., "Constrained Admission Control to a Queueing System," *Advances in Applied Probability*, **21**, 409–431 (1989).

[4] Kaufman, J.S., "Blocking in a Shared Resource Environment," *IEEE Transactions on Communications*, **COM-29**, 1474–1481 (1981).

[5] Kelly, F.P., *Reversibility and Stochastic Networks*, Wiley, New York, 1979.

[6] Lam, S.S., "Queueing Networks with Population Size Constraints," *IBM Journal of Research and Development*, **21**, 370–378 (1977).

[7] Ma, D.-J., and Makowski, A.M., "Optimality Results for a Simple Flow Control Problem," in *Proc. of the 26th Conf. on Decision and Control*, IEEE Press, Piscataway, NJ, 1987, pp. 1852–1857.

[8] Nain, P., and Ross, K.W., "Optimal Multiplexing of Heterogeneous Traffic with Hard Constraint," *Performance Evaluation Review*, **14**, 100–108 (1986).

[9] Nain, P., and Ross, K.W., "Optimal Priority Assignment with Hard Constraint," *IEEE Transactions on Automatic Control*, **AC-31**, 883–888 (1986).

[10] Reiser, M., and Kobayashi, H., "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," IBM Journal of Research and Development, **19**, 283–294 (1975).

[11] Robertazzi, T.G., and Lazar, A.A., "On the Modeling and Optimal Flow Control of the Jacksonian Network," *Performance Evaluation*, **5**, 29–43 (1985).

[12] Ross, K.W., "Constrained Markov Decision Processes with Queueing Applications," Ph.D. thesis, Computer, Information and Control Engineering, University of Michigan, 1985.

[13] Ross, K.W., and Tsang, D.H.K., "The Stochastic Knapsack Problem," *IEEE Transactions on Communications*, **37**, 740–747 (1989).

[14] Ross, K.W., and Yao, D.D., "Monotonicity Properties for the Stochastic Knapsack," *IEEE Transactions on Information Theory*, **36**, 1173–1179 (1990).

[15] de Waal, P.R., "Overload Control of Telephone Exchanges," Ph.D. thesis, CWI, Amsterdam, 1990.

[16] de Waal, P.R., and van Dijk, N.M., "Monotonicity of Performance Measures in a Processor Sharing Queue," *Performance Evaluation*, **12**(1), 5–16 (1991).

[17] de Waal, P.R., and van Dijk, N.M., "Interconnected Networks of Queues with Randomized Arrival and Departure Blocking," *Annals of Operations Research*, **35**, 97–124 (1992).

Manuscript received August 13, 1991
Revised manuscript received January 2, 1993
Accepted February 18, 1993