Probability in the Engineering and Informational Sciences, **16**, 2002, 289–307. Printed in the U.S.A.

OPTIMAL CONTROL OF PARALLEL QUEUES WITH BATCH SERVICE

CATHY H. XIA
IBM T.J. Watson Research Center
Yorktown, NY 10598
E-mail: cathyx@us.ibm.com

GEORGE MICHAILIDIS
Department of Statistics
University of Michigan
Ann Arbor, MI
E-mail: gmichail@umich.edu

NICHOLAS BAMBOS
Departments of Electrical Engineering and
Management Science & Engineering
Stanford University
Stanford, CA
E-mail: bambos@stanford.edu

PETER W. GLYNN
Department of Management Science & Engineering
Stanford University
Stanford, CA
E-mail: glynn@leland.stanford.edu

We consider the problem of dynamic allocation of a single server with *batch processing* capability to a set of parallel queues. Jobs from different classes cannot be processed together in the same batch. The arrival processes are mutually independent Poisson flows with equal rates. Batches have independent and identically distributed exponentially distributed service times, independent of the batch size and the arrival processes. It is shown that for the case of infinite buffers, allocating the server to the longest queue, stochastically maximizes the aggregate throughput of the system. For the case of equal-size finite buffers the same policy stochastically minimizes the loss of jobs due to buffer overflows. Finally, for the case of *unequal*-size buffers, a threshold-type policy is identified through an extensive simulation study and shown to consistently outperform other conventional policies. The good performance of the proposed threshold policy is confirmed in the heavy-traffic regime using a fluid model.

© 2002 Cambridge University Press 0269-9648/02 $12.50

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

1. THE MODEL

We consider the problem of dynamic allocation of a single server with *batch processing* capability to a set of parallel queues. Jobs from different classes cannot be processed together in the same batch. A canonical application of the model is in the area of flexible manufacturing where a factory workstation has batch-processing capabilities and is able to work on multiple classes of production processes (e.g., an oven in a semiconductor manufacturing plant). However, each production process has its own set of specifications, therefore prohibiting the mixing of jobs from different processes. Other applications can also be found in transportation systems or communication systems. A typical example in communication systems is a serial-to-parallel bus, where messages arrive to some serial input port and need to be forwarded to B slower output ports. The batch capability of the server captures this essential feature of such a device. In all of these applications, important performance measures are the system throughput and, in the presence of finite buffers, the job loss due to overflows.

To make things concrete about the model, we consider a system composed of N parallel queues served by a single server. The queues have buffer sizes $C_i, i = 1, \ldots, N$, that could be infinite or finite. The job arriving stream to the ith queue is a Poisson process $\mathbf{A}_i = \{A_i(t); t \in \mathbb{R}^+\}$, where $A_i(t)$ is the number of attempted arrivals to queue i in the time interval $[0, t)$. The arrival processes $\mathbf{A}_1, \ldots, \mathbf{A}_N$ are assumed to be mutually independent and have equal rates λ (symmetric arrivals). We next specify the batching mechanism employed by the server:

- Each batch must be formed from jobs belonging to the same queue.
- The maximum batch size is B jobs, where $B < C_i$, $i = 1, \ldots, N$; however, the server is allowed to pick any number of jobs $\leq B$ (from the same queue) when forming a batch.
- Batches have independent and identically distributed (i.i.d.) exponentially distributed service times, independent of the batch size and the arrival processes.

Finally, it is assumed that the resetting cost of the server for switching to a different class is essentially zero.

At certain time instants the server must decide from which of the N queues to form a batch for processing. The decision mechanism employed by the server defines a *server allocation policy*. In this article we restrict attention to the set Γ of *nonanticipative*, *nonpreemptive*, and *nonidling* policies. The nonanticipative restriction implies that the server utilizes only present and past system state information when making decisions. We are interested in dynamically allocating the server so as to stochastically maximize the system's throughput and, in the case of finite buffers, to minimize the long-term average loss flow (due to buffer overflows).

The parallel queuing scheduling problem for a single server with *single job processing* capability has received considerable attention in the literature. For perfectly symmetric systems (equal rate Poisson arrivals, i.i.d. exponential service times, and equal-size buffers), it has been shown in previous articles [16,18] that the Long-

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

est Queue first (LQ) policy stochastically maximizes the total departure process and, in the case of finite buffers, stochastically minimizes the number of total job losses due to overflow among all preemptive and nonidling policies. The result has been extended in [17] to the case of unequal buffers and general i.i.d. service distribution. By using a coupling argument, it has been shown that the fewest empty space first (FES) policy stochastically minimizes the total-loss process among all nonpreemptive and nonidling policies.

On the other hand, the problem of *batch processing* for a single queue with Poisson arrival and general renewal service processes (independent of the batch size) has been studied in [2], where Markov decision theory is used to establish the optimality of a threshold-type policy with idling allowed. A similar problem is considered in [1], where the batch-processing time is a function of the individual jobs' processing times. It is shown that a threshold-type policy optimizes a long-run average-cost criterion. In [4], structural properties of the optimal policy that minimizes the makespan and the flow time are derived for the above problem. In [3], the static (no future arrivals) and dynamic problems of the optimal scheduling of incompatible classes of jobs on a batch-processing machine are examined, and a characterization of the optimal policy is provided along with an easy-to-implement heuristic scheduling scheme. Other heuristic scheduling policies for the static problem are presented in [15].

The batch capability of the server fundamentally alters the queuing dynamics of the system and introduces a dilemma not present in the case of single job processing. This can be best explained by considering the following example. In a system comprised of two parallel queues with buffer capacity $C_1 = 5$, $C_2 = 10$ and maximum batch size $B = 4$, suppose that at some decision instant, the queue lengths are 3 and 5, respectively. If the server decides to serve the first queue because it is closer to overflowing, it would result in an underutilization of resources. If, on the other hand, it decides to serve the second queue, it risks losing jobs due to a possible overflow of the first queue. Thus, in general, the server needs to strike a fine balance between these two objectives. For systems with single job processing capabilities, this is not an issue because no matter which queue the server is allocated to, it can only process one job.

In this article, we show that, for the case of infinite buffers, allocating the server to the longest queue stochastically maximizes the aggregate throughput of the system. For the case of equal-size finite buffers, the same policy stochastically minimizes the loss of jobs due to buffer overflows. Finally, for the case of *unequal*-size buffers, a threshold-type policy is identified through an extensive simulation study and shown to consistently outperform other conventional policies. The good performance of the proposed threshold policy is confirmed in the heavy-traffic regime using a fluid model.

To show the results we make extensive use of the coupling method to establish desired stochastic ordering [6]. These techniques have proved to be powerful tools in solving optimal scheduling or routing problems (e.g., see [5,7,9,11–13] and the references cited therein). Building on this basis, this article contributes in the following two directions. First, the classical parallel queuing model is extended by

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

adding the batch-processing capability of the server. Second, the presence of a batch service mechanism introduces complications both at the technical and at the conceptual levels in making pathwise comparisons between alternative policies under various coupling structures. Moreover, the case of unequal buffers introduces new sample path dynamics that are not present in the single job processing capability case. The emerging complications are then successfully resolved.

The remainder of this article is organized as follows. Section 2 covers the notations and some preliminary results. Sections 3 and 4 present the detailed proofs using coupling arguments for systems with infinite buffer capacities and for systems with equal-size finite buffers, respectively. In Section 5, the case of unequal-size finite buffers is discussed. We present selected simulations that help identify a threshold-type policy, which is then shown to be optimal using a fluid model approach in heavy traffic. Finally, concluding remarks are given in Section 6.

2. NOTATIONS AND PRELIMINARIES

Before proceeding with the results we define some notations that will be used later. For a system operating under server allocation policy $\gamma \in \Gamma$, let $\{\mathbf{X}^\gamma(t); t \geq 0\}$ denote the *joint queue length* process, where $\mathbf{X}^\gamma(t) = (X_1^\gamma(t), \ldots, X_N^\gamma(t))$, with $X_i^\gamma(t)$ being the number of jobs in queue i (not including the jobs in service) at time t. Define the *total-loss* process $\{L^\gamma(t); t \geq 0\}$, where $L^\gamma(t)$ gives the total number of lost jobs due to buffer overflows in the time interval $[0, t)$ for policy $\gamma \in \Gamma$. Similarly, define the *total-departure* process $\{D^\gamma(t); t \geq 0\}$, where $D^\gamma(t)$ gives the total number of jobs that have departed from the system in $[0, t)$. Finally, denote by $b^\gamma(t)$ the batch size that policy γ allocates to the server at decision instant t.

In addition, we introduce the following partial ordering of random variables and processes. More details on stochastic orderings can be found in [11,13] for example.

DEFINITION 2.1: *Given two random vectors $X, Y \in \mathbb{R}^n$, X is said to be stochastically smaller than Y ($X \leq_{\text{st}} Y$) if $E[f(X)] \leq E[f(Y)]$ for all increasing $f: \mathbb{R}^n \mapsto \mathbb{R}$.*

Given two stochastic processes $\{X(t)\}$ and $\{Y(t)\}$, $t \in \mathbb{R}_+$, the process $\{X(t)\}$ is said to be stochastically smaller than $\{Y(t)\}$, ($\{X(t)\} \leq_{\text{st}} \{Y(t)\}$) if $\forall\ n \in \mathbb{N}$, $\forall\ (t_1, t_2, \ldots, t_n) \in \mathbb{R}_+^n$,

$$(X(t_1), X(t_2), \ldots, X(t_n)) \leq_{\text{st}} (Y(t_1), Y(t_2), \ldots, Y(t_n)).$$

The next two lemmas identify some important properties of the scheduling policies which are needed later.

LEMMA 2.2: *Given two initial states $\mathbf{a}(0) = (a_1(0), \ldots, a_N(0))$ and $\mathbf{b}(0) = (b_1(0), \ldots, b_N(0))$, suppose that*

$$b_i(0) \leq a_i(0), \qquad i = 1, \ldots, N. \tag{1}$$

For an arbitrary policy $\alpha \in \Gamma$ that operates on a system starting with $\mathbf{a}(0)$, there exists a policy $\beta \in \Gamma$ which starts with $\mathbf{b}(0)$, such that

$$\{\mathbf{X}^\beta(t); t \geq 0\} \leq_{st} \{\mathbf{X}^\alpha(t); t \geq 0\}, \tag{2}$$

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

where $\mathbf{X}^{\alpha}(t)$ *denotes the* queue *length vector of the system operating under policy* α *at time* t*, and similarly for* $\mathbf{X}^{\beta}(t)$*. When all buffers are of finite sizes, this implies*

$$\{L^{\beta}(t); t \geq 0\} \leq_{\text{st}} \{L^{\alpha}(t); t \geq 0\}.$$

PROOF: In order to show (2), we simply need to construct a coupling $(\hat{X}^{\alpha}, \hat{X}^{\beta})$ of the random processes (X^{α}, X^{β}), such that $\hat{X}^{\alpha} \stackrel{d}{=} X^{\alpha}, \hat{X}^{\beta} \stackrel{d}{=} X^{\beta}$, and for all $t \geq 0$,

$$\hat{X}_i^{\beta}(t) \leq \hat{X}_i^{\alpha}(t), \qquad i = 1, \ldots, N, \tag{3}$$

and in the finite buffer case,

$$\hat{L}^{\beta}(t) \leq \hat{L}^{\alpha}(t), \tag{4}$$

where the "hat" symbol denotes the coupled versions of the processes, and $\stackrel{d}{=}$ denotes the equality of their finite dimensional distributions. Refer to, for example, [6] for details about coupling and stochastic comparison techniques.

Because all of the arrival processes are Poisson of equal rate λ, one can uniformize the input flows and assume equivalently that there exists a single Poisson input process of rate $N\lambda$, and every arrival can join each of the N queues with equal probability $1/N$.

We now couple the system operating under policy β, with the one operating under policy α in the following way. First, we give the two systems the same arrival processes. Second, letting $\hat{S}_n^{\alpha}$ be the service time of the nth batch to complete service under α, we set $\hat{S}_n^{\beta} = \hat{S}_n^{\alpha}$ (since the service times are i.i.d. exponentially distributed, independent of the batch sizes, and, hence, with identical statistics in all queues). For ease of exposition, we denote queue i in the system operating under policy α by i^{α}, and similarly we define index i^{β}.

Note that the queue length vector will only change at an arrival epoch or at a decision epoch. Because the arrival processes of both systems are perfectly synchronized, at an arrival epoch the same queue in both systems will simultaneously increase by 1; in the finite buffer case, the arrival may be lost by both systems or only by the system under α, given relation (3) holds before the arrival. In both cases, relation (3) continues to hold (inductively).

Suppose (3) holds for $t \in [0, \tau]$, where $\tau \geq 0$ is a decision epoch under policy α. At this instant, suppose policy α allocates the server to queue k and serves a batch of size $b^{\alpha}(\tau)$. We examine next the following three cases.

If $X_k^{\beta}(\tau) > 0$, then let policy β also allocate the server to queue k and serve a batch of size $b^{\beta}(\tau) = b^{\alpha}(\tau) \wedge X_k^{\beta}(\tau)$, where $a \wedge b := \min\{a, b\}$; that is, policy β is mimicking α as much as possible.

If $X_k^{\beta}(\tau) = 0$ and there exist some other buffers with $X_i^{\beta}(\tau) > 0$, then let β choose arbitrarily such a queue i and serve a batch of arbitrary size $0 < b^{\beta}(\tau) \leq B \wedge X_i^{\beta}(\tau)$.

If all $X_i^{\beta}(\tau) = 0$, then policy β does nothing until the next arrival. If upon this arrival, policy α is still serving a batch, and because β needs to be nonidling, we then let β immediately start a batch of size 1 with a service time equal to the remaining service time of the batch being served by policy α. Note that this is feasible because

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

the batch service time is exponentially distributed, and therefore the random variable $S - u | S > u$ is still exponentially distributed.

Under the above construction, it is easy to check that (3) continues to hold from τ^+ until the next decision epoch.

By inductively applying the above argument, policy β is then constructed in a pathwise fashion with relationship (3) holding for all times, where β is clearly nonidling, nonpreemptive, and nonanticipative. Because losses only occur at arrival epochs, as long as (3) holds, (4) follows easily. This completes the proof of the lemma. ■

Note that the scheduling decision is twofold: first, to decide which queue to serve next and, then, how many jobs to be included in a batch. The next lemma addresses the second issue and establishes the optimal batch size. It is based on the observation that a larger batch is always preferred because service times are independent of the batch size; that is, at each decision epoch, once the queue to serve is determined, say queue i, the server should always form a batch of size

$$B \wedge X_i, \tag{5}$$

where X_i is the number of jobs present in queue i. We denote by Γ^* ($\subset \Gamma$) the set of all policies that follow this batching rule.

LEMMA 2.3: *For an arbitrary policy $\beta \in \Gamma$, there exists a policy $\beta^* \in \Gamma^*$ such that*

$$\{X^{\beta^*}(t); t \geq 0\} \leq_{\text{st}} \{X^{\beta}(t); t \geq 0\}. \tag{6}$$

When all buffers are finite, this implies

$$\{L^{\beta^*}(t); t \geq 0\} \leq_{\text{st}} \{L^{\beta}(t); t \geq 0\};$$

hence, relationship (5) gives the optimal batching rule.

PROOF: Set $\beta_0 = \beta$. If $\beta_0 \in \Gamma^*$, then the proof is complete.

If $\beta_0 \notin \Gamma^*$, let t_0 be the first decision instant that it does not follow (5) when forming a batch. Suppose at that instant that the server is allocated to queue j. We then construct a coupled system by giving it the same arrival and service processes (as in the proof of Lemma 2.2) and define policy β_1 as follows: During time interval $[0, t_0)$, let β_1 follow β_0; at decision epoch t_0, let β_1 allocate the server also to queue j, but form a batch according to (5). This then gives

$$\hat{X}_j^{\beta_1}(t_0^+) \leq \hat{X}_j^{\beta_0}(t_0^+),$$

$$\hat{X}_i^{\beta_1}(t_0^+) = \hat{X}_i^{\beta_0}(t_0^+).$$

Now, simply apply Lemma 2.2 to define the actions of β_1 for $t > t_0$. We then have

$$\{X^{\beta_1}(t); t \geq 0\} \leq_{\text{st}} \{X^{\beta_0}(t); t \geq 0\}.$$

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

By inductively applying the above argument, we can improve the policies sequentially. Clearly, the limit β^* is a policy that follows (5) at each event instant; thus, it belongs to Γ^*. ■

Given the result of Lemma 2.3, we can now simply focus on the problem to which queue to allocate the processing power of the server while assuming that the server always obeys the optimal batching rule given by (5).

3. THE CASE OF INFINITE BUFFERS

In this section we prove the optimality of the Longest Queue (LQ) first policy for the case of infinite buffers.

PROPOSITION 3.1: *The Longest Queue first policy (allocating the server to the longest queue and forming a batch as large as possible) stochastically minimizes the total backlog and maximizes the aggregate throughput of a system with infinite buffers; that is,*

$$\left\{\sum_i X_i^{\mathrm{LQ}}(t); t \geq 0\right\} \leq_{\mathrm{st}} \left\{\sum_i X^{\gamma}(t); t \geq 0\right\}$$

and

$$\{D^{\mathrm{LQ}}(t); t \geq 0\} \geq_{\mathrm{st}} \{D^{\gamma}(t); t \geq 0\}$$

for all nonpreemptive and nonidling $\gamma \in \Gamma$, with $X^{\mathrm{LQ}}(0) = X^{\gamma}(0)$.

PROOF: From Lemma 2.3 we can simply restrict our attention to policies in Γ^*. Let $\gamma \in \Gamma^*$ be an arbitrary policy. Without loss of generality, let $t = 0$ be the first decision instant that policy γ differs from the LQ policy. It then suffices to show that there exists a policy $\pi \in \Gamma$, which follows LQ at $t = 0$ (and is appropriately defined at all other decision instants), such that

$$\left\{\sum_i X_i^{\pi}(t); t \geq 0\right\} \leq_{\mathrm{st}} \left\{\sum_i X_i^{\gamma}(t); t \geq 0\right\} \tag{7}$$

and

$$\{D^{\pi}(t); t \geq 0\} \geq_{\mathrm{st}} \{D^{\gamma}(t); t \geq 0\} \tag{8}$$

provided that $X^{\pi}(0) = X^{\gamma}(0)$. Based on Lemma 2.3, we can then find an improved policy $\pi^* \in \Gamma^*$ that satisfies (7) as well. The proposition can then be established by inductively applying the above argument.

In order to show (7) and (8), again we use the coupling argument. We construct two coupled (hat) systems: one operating under γ and the other under π, where $\hat{X}^{\pi}(0) = \hat{X}^{\gamma}(0)$, and they have the same arrival and service processes as described in the proof of Lemma 2.2. Let $\hat{X}^{\pi}$, $\hat{X}^{\gamma}$, and $\hat{D}^{\pi}$, $\hat{D}^{\gamma}$ be the corresponding queue lengths and departure processes, respectively, under this construction.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

Let $\{A^\gamma(t); t \geq 0\}$ be the total number of arrivals in the time interval $[0, t)$ in the system under policy γ. Based on the structural relations

$$\hat{X}^\xi(0) + \hat{A}^\xi(t) = \sum_{i=1}^{N} \hat{X}_i^\xi(t) + \hat{D}^\xi(t), \qquad \xi = \gamma, \pi,$$

it can be seen that (8) follows immediately from (7) due to the coupling of the initial queue lengths and the arrival processes of the two systems. Hence, it suffices to establish that

$$\sum_{i=1}^{N} \hat{X}_i^\pi(t) \leq \sum_{i=1}^{N} \hat{X}_i^\gamma(t) \tag{9}$$

for all $t \geq 0$.

Suppose that queue l has the maximum number of jobs at time $t = 0$. Then policy π allocates the server to queue l, serving a batch of size $b^\pi(0) = \hat{X}_l^\pi(0) \wedge B$, whereas γ assigns the server to some other nonempty queue j with $\hat{X}_j(0) < \hat{X}_l(0)$, serving a batch of size $b^\gamma(0) = \hat{X}_j(0) \wedge B$ (because $\gamma \in \Gamma^*$). The queue lengths immediately after time $t = 0$, in the systems operating under policies π and γ respectively, are then given by

$$\hat{X}_l^\pi(0^+) = \hat{X}_l^\gamma(0^+) - b^\pi(0), \tag{10}$$

$$\hat{X}_j^\pi(0^+) = \hat{X}_j^\gamma(0^+) + b^\gamma(0), \tag{11}$$

$$\hat{X}_i^\pi(0^+) = \hat{X}_i^\gamma(0^+), \quad i \neq j, l. \tag{12}$$

We now consider the following two cases.

Case a: Suppose $\hat{X}_l(0) > B$; then, $b^\pi(0) = B$. From $t = 0^+$ on, let policy π follow policy γ *exactly* (i.e., serving the same queue with the same batch size) until the first time τ it cannot do so; that is, τ is the first decision instant that γ allocates the server to queue l to serve a batch of size $b^\gamma(\tau) > \hat{X}_l^\pi(\tau)$. If there is no such instant, we then simply set $\tau = \infty$.

In the case $\tau = \infty$, relation (9) follows immediately, as π has served more jobs than γ in the first decision epoch and then follows γ exactly afterward.

Now suppose $\tau < \infty$. Because π follows γ exactly in the time interval $(0, \tau)$, relations (10)–(12) continue to hold. At time $t = \tau$, let π allocate the server to queue j and serve a batch of size $b^\gamma(0)$. We then get

$$\hat{X}_l^\pi(\tau^+) = \hat{X}_l^\gamma(\tau^+) - b^\pi(0) + b^\gamma(\tau) \leq \hat{X}_l^\gamma(\tau^+), \tag{13}$$

$$\hat{X}_j^\pi(\tau^+) = \hat{X}_j^\gamma(\tau^+), \tag{14}$$

$$\hat{X}_i^\pi(\tau^+) = \hat{X}_i^\gamma(\tau^+), \qquad i \neq j, l. \tag{15}$$

Now simply apply Lemma 2.2 to construct actions of policy π for $t > \tau$; the results of Lemma 2.2 then imply (9) immediately.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

Case b: Suppose $\hat{X}_l(0) \le B$. We then have that $b^\pi(0) = \hat{X}_l(0) > \hat{X}_j(0) = b^\gamma(0)$. The queue lengths of both systems after the allocation are then given by

$$\hat{X}_l^\pi(0^+) = 0 = \hat{X}_j^\gamma(0^+),$$
$$\hat{X}_j^\pi(0^+) = b^\gamma(0) < b^\pi(0) = \hat{X}_l^\gamma(0^+),$$
$$\hat{X}_i^\pi(0^+) = \hat{X}_i^\gamma(0^+), \qquad i \neq j, l.$$

We now switch the roles of queues l and j in the system operating under policy π; that is, queue l (resp. j) in the system operating under policy π will be coupled with queue j (resp. l) in the system operating under policy γ. The above switch implies that whenever there is an arrival to l^γ (resp. j^γ), there is also an arrival to queue j^π (resp. l^π). This is permissible because each arrival of the uniformized Poisson input stream is equally likely to join each of the N queues. The switch then gives

$$\hat{X}_{[j]}^\pi(0^+) = \hat{X}_j^\gamma(0^+), \tag{16}$$
$$\hat{X}_{[l]}^\pi(0^+) < \hat{X}_l^\gamma(0^+), \tag{17}$$
$$\hat{X}_{[i]}^\pi(0^+) = \hat{X}_i^\gamma(0^+), \qquad i \neq j, l, \tag{18}$$

where the subscripts $[i]$, $i = 1, \ldots, N$, denote the new labels of the buffers in the system operating under policy π.

Now simply apply Lemma 2.2 to construct actions of policy π for $t > 0$; the results of Lemma 2.2 then imply (9) immediately.

In both cases, we have established that $\sum_i \hat{X}_i^\pi(t) \le \sum_i \hat{X}_i^\gamma(t)$ for all $t \ge 0$. This completes the proof of the proposition. ■

4. OPTIMALITY OF FES TO FINITE BUFFERS WITH EQUAL SIZES

In this section, we establish the optimality of the Fewest Empty Space (FES) first policy for the case of equal-size finite buffers. In particular, we show that it minimizes the average long-run loss of jobs due to buffer overflows. It is worth noting that due to the fact that all queues have equal buffer capacity, the FES policy is equivalent to the LQ policy.

PROPOSITION 4.1: *For systems with equal-size finite buffers, the FES policy (allocating the server to the Fewest Empty Space queue first and forming a batch as large as possible) stochastically minimizes the average long-run loss of jobs due to buffer overflows; that is,*

$$\{L^{\mathrm{FES}}(t); t \ge 0\} \le_{\mathrm{st}} \{L^\gamma(t); t \ge 0\} \tag{19}$$

for all nonpreemptive and nonidling $\gamma \in \Gamma$ *provided* $X^{FES}(0) = X^\gamma(0)$.

PROOF: From Lemma 2.3, we can simply restrict our attention to policies in Γ^*. Let $\gamma \in \Gamma^*$ be an arbitrary policy and let $t = 0$ be the first decision instant that it differs from the FES policy (if there is no such instant, then the proof is complete.). It

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

suffices to show that there exists a policy $\pi \in \Gamma$ which follows policy FES at $t = 0$ (and is appropriately defined afterward) such that

$$\{L^{\pi}(t); t \geq 0\} \leq_{\text{st}} \{L^{\gamma}(t); t \geq 0\}. \tag{20}$$

Based on Lemma 2.3, we can then find an improved policy $\pi^* \in \Gamma^*$ also satisfying (20). By inductively applying the argument, we then establish the proposition.

In order to show (20), we will again make use of the coupling argument. First assume that $\hat{X}^{\pi}(0) = \hat{X}^{\gamma}(0)$. We couple the arrival processes (resp. the service processes) of the two systems operating under policies γ and π in the same way as in the proof of Lemma 2.2. In addition, define η_i^{γ} (resp. η_i^{π}) to be the first time that buffer i^{γ} (resp. i^{π}) overflows.

At $t = 0$, suppose that queue l is the longest. However, policy γ allocates the server to some other nonempty queue j with $X_j(0) < X_l(0)$ and forms a batch of size $b^{\gamma}(0) = B \wedge \hat{X}_j(0)$ (because $\gamma \in \Gamma^*$) (see Fig. 1). Let policy π allocate the server to queue l instead and form a batch of size $b^{\pi}(0) = B \wedge \hat{X}_l(0)$. Clearly, $b^{\pi}(0) \geq b^{\gamma}(0)$, and the queue length relationships between the two systems immediately after time 0 are given by

$$\hat{X}_l^{\pi}(0^+) = \hat{X}_l^{\gamma}(0^+) - b^{\pi}(0), \tag{21}$$

$$\hat{X}_j^{\pi}(0^+) = \hat{X}_j^{\gamma}(0^+) + b^{\gamma}(0), \tag{22}$$

$$\hat{X}_i^{\pi}(0^+) = \hat{X}_i^{\gamma}(0^+), \qquad i \neq j, l. \tag{23}$$

We now consider the following two cases: (I) $X_l(0) > B$ and (II) $X_l(0) \leq B$.

Case I: Suppose that $X_l(0) > B$, which immediately implies that $b^{\pi}(0) = B$. We must then have for $t = 0$,

$$\hat{X}_j^{\gamma}(t^+) < \hat{X}_l^{\pi}(t^+), \tag{24}$$

$$C - \hat{X}_l^{\gamma}(t^+) < C - \hat{X}_j^{\pi}(t^+). \tag{25}$$

Note that as long as relations (24) and (25) are maintained, buffer j^{π} will not overflow and the total overflow of the system under policy π cannot be larger than that of the system under policy γ.

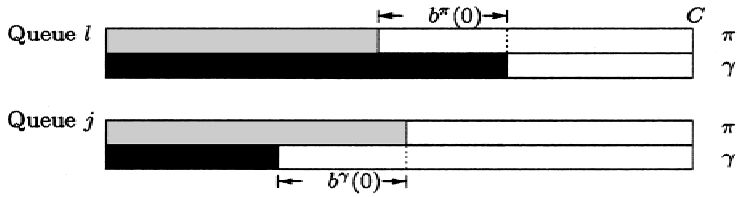


FIGURE 1. Queue lengths at $t = 0^+$.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

For $t > 0$, let policy π follow policy γ (serving the same queue with the same batch size) until time $T := \min(\sigma, \zeta, \tau)$, where σ, ζ, and τ are defined as follows:

a. Let σ be the first time that π can no longer come up with the same batch size as γ. At this instant, policy γ must be serving queue l with batch size $b^\gamma(t) > X_l^\pi(t)$.
b. Let ζ be the first time that γ serves queue l after η_l^γ (i.e., after buffer l^γ overflowed).
c. Let τ be the first time that (25) no longer holds. This could be either because more jobs arrived at buffer j or because more service effort was allocated to queue l.

To define the actions of policy π for $t \geq T$, we discuss the three cases separately.

Case a: Suppose that $T = \sigma$. We must have $\sigma < \eta_l^\gamma$, otherwise σ overlaps with ζ (which is covered by Case b). Because inequality (25) holds during $(0, T)$, we also have $\sigma < \eta_j^\pi$. Hence, during $(0, T)$, there are no overflows from buffer l or buffer j in either system, and policy π has followed γ exactly. Therefore, relations (21)–(23) continue to hold in this period. Because $\gamma \in \Gamma^*$, relation (21) implies that $b^\gamma(T) = B$ and $X_l^\pi(T) < B$.

At time T, let policy π allocate the server to queue j instead and serve a batch of size $b^\gamma(0)$. We then have $\hat{X}_i^\pi(T^+) = \hat{X}_i^\gamma(T^+)$, for all $i = 1, \ldots, N$. For $t > T$, let π follow γ and the two systems are essentially the same. It then follows immediately that $\hat{L}^\pi(t) = \hat{L}^\gamma(t)$ for all $t \geq 0$.

Case b: Suppose that $T = \zeta$. This implies that $X_l^\gamma(T) = C$; thus, $b^\gamma(T) = B$. In addition, because (25) holds during $(0, T)$, it follows that $\eta_j^\pi > T$. Therefore, there is no overflow from buffer j in either of the two systems and (22) continues to hold during $(0, T)$.

Let $\Delta\hat{L}_l(t) := \hat{L}_l^\gamma(t) - \hat{L}_l^\pi(t)$ be the difference between the total number of lost jobs from buffer l under the two policies in the time interval $(0, t)$. From (21), we then have

$$\hat{X}_l^\pi(T) = \hat{X}_l^\gamma(T) - b^\pi(0) + \Delta\hat{L}_l(T). \tag{26}$$

Clearly, $0 \leq \Delta\hat{L}_l(t) \leq B$ during $(0, T]$, and $\hat{X}_l^\pi(t) \leq \hat{X}_l^\gamma(t)$ in this time interval.

At time T, let policy π allocate the server to queue j instead and serve a batch of size $b^\gamma(0)$. From (26) and (22), we then have

$$\begin{aligned} \hat{X}_l^\pi(T^+) &= \hat{X}_l^\gamma(T^+) + \Delta\,\hat{L}_l\,(T), \\ \hat{X}_j^\pi(T^+) &= \hat{X}_j^\gamma(T^+), \\ \hat{X}_i^\pi(T^+) &= \hat{X}_i^\gamma(T^+), \qquad i \neq j,\, l. \end{aligned} \tag{27}$$

For $t > T$, let π follow policy γ exactly. Because queue l^π is no less than queue l^γ, π should always be able to come up with the same batch size as γ. Note that from

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

T on, buffer l^π would lose at most $\Delta\hat{L}_l(T)$ more jobs than buffer l^γ. Therefore, $\hat{L}_l^\pi(t) \le \hat{L}_l^\gamma(t)$ for all $t \ge 0$, and the overflow from other buffers are the same.

Case c: Suppose that $T = \tau$. Note that time τ could correspond to either an arrival epoch for queue j or a decision instant for queue l under policy γ. In the latter case, we would have that $\hat{X}_l^\gamma(\tau) - b^\gamma(\tau) \le \hat{X}_j^\pi(\tau)$, which would destroy relation (25) at time τ^+. We examine the two cases separately.

Case c.1: Suppose that τ is a decision instant. We must have $\eta_l^\gamma > \tau$, otherwise ζ overlaps with τ (which is covered by Case b). In addition, $\eta_j^\pi > \tau$ because (25) holds during $(0, \tau)$. Therefore, (21)–(23) continue to hold during this time interval and $b^\gamma(\tau) = B$ because $\gamma \in \Gamma^*$.

At $t = \tau$, let policy π allocate the server to queue j and serve a batch of size $b^\gamma(0)$. It then follows that $\hat{X}_i^\pi(\tau^+) = \hat{X}_i^\gamma(\tau^+)$, for all $i = 1, \ldots, N$. For $t > \tau$, let π follow γ exactly and the two systems are essentially the same. Thus, $\hat{L}^\pi(t) = \hat{L}^\gamma(t)$ for all $t \ge 0$.

Case c.2: Suppose that τ is an arrival instant. Because τ is the first instant that buffer j^π reaches the same empty space level (upon arrival) as buffer l^γ, regardless of whether this arrival occurs before time η_l^γ or after, we must have $\tau < \eta_j^\pi$. Thus, relation (22) must hold for queue j at time τ. We then have

$$\hat{X}_l^\gamma(\tau^+) = \hat{X}_j^\pi(\tau^+), \tag{28}$$

$$\begin{aligned}\hat{X}_j^\gamma(\tau^+) &= \hat{X}_j^\pi(\tau^+) - b^\gamma(0), \\ &\ge \hat{X}_l^\gamma(\tau^+) - b^\pi(0) = \hat{X}_l^\pi(\tau^+) - \Delta\hat{L}_l(\tau),\end{aligned} \tag{29}$$

where, in (29), the inequality is because (28) and $b^\pi(0) \ge b^\gamma(0)$, whereas the last equality is directly from (26).

From time τ^+ on, we switch the roles of queues l and j in the system operating under policy π in the coupling, so that queue l (resp. j) in the system under π will be coupled with queue j (resp. l) in the system under γ for both the arrival and service processes; that is, whenever there is an arrival to queue l (resp. j) in the system under γ, there is also an arrival to queue j (resp. l) in the system under π; similarly for the service processes. The switch then gives

$$\hat{X}_l^\gamma(\tau^+) = \hat{X}_{[l]}^\pi(\tau^+),$$

$$\hat{X}_j^\gamma(\tau^+) \ge \hat{X}_{[j]}^\pi(\tau^+) - \Delta\hat{L}_{[j]}(\tau), \tag{30}$$

$$\hat{X}_i^\gamma(\tau^+) = \hat{X}_{[i]}^\pi(\tau^+), \qquad i \ne j, l, \tag{31}$$

where the subscripts $[i]$, $i = 1, \ldots, N$, denote the new labels of the buffers in the system operating under policy π, and $\Delta\hat{L}_{[j]}(\tau) := \Delta\hat{L}_l(\tau) \in [0, B]$.

In (30), if $\hat{X}_j^\gamma(\tau^+) \ge \hat{X}_{[j]}^\pi(\tau^+)$, simply apply Lemma 2.2 to construct π for $t > \tau$, which then gives $\hat{X}^\pi(t) \le \hat{X}^\gamma(t)$ for all $t > \tau$. It then follows immediately that $\hat{L}^\pi(t) \le \hat{L}^\gamma(t)$ for all $t \ge 0$.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

If $\hat{X}_j^{\gamma}(\tau^+) < \hat{X}_{[j]}^{\pi}(\tau^+)$, then policy π can always follow γ exactly for $t > \tau$. From time τ and afterward, queue $[j]$ in the system under π will lose at most $\Delta\hat{L}_{[j]}(\tau)$ more jobs than γ. Therefore, $\hat{L}^{\pi}(t) \le \hat{L}^{\gamma}(t)$ for all $t \ge 0$.

Case II: Suppose that $b^{\pi}(0) \le B$. We then have that $b^{\gamma}(0) = \hat{X}_j(0) < \hat{X}_l(0) = b^{\pi}(0)$. Immediately after time $t = 0$, the queue lengths of both systems are given by

$$\hat{X}_j^{\gamma}(0^+) = \hat{X}_l^{\pi}(0^+) = 0,$$

$$\hat{X}_l^{\gamma}(0^+) = b^{\pi}(0) > b^{\gamma}(0) = \hat{X}_j^{\pi}(0^+),$$

$$\hat{X}_i^{\gamma}(0^+) = \hat{X}_i^{\pi}(0^+), \qquad i \ne j, l.$$

This can be considered simply as a special case of Case c.2. We just switch the role of queue l and queue j in the system operating under π, so that queue l^{π} (resp. j^{π}) will be coupled with queue j^{γ} (resp. l^{γ}) for both the arrival and service processes. This then gives relation (1) at time 0^+. Again apply Lemma 2.2; it immediately follows that $\hat{L}^{\pi}(t) \le \hat{L}^{\gamma}(t)$ for all $t \ge 0$.

Hence, in all cases, we have $\hat{L}^{\pi}(t) \le \hat{L}^{\gamma}(t)$ for all $t \ge 0$. This completes the proof of the proposition. ■

5. FINITE BUFFERS OF UNEQUAL SIZES

In many practical situations, the buffers of the various queues may have unequal sizes. For obvious reasons, we assume that $\min_i\{C_i\} > B$. As mentioned in Section 1, Wasserman and Bambos [17] showed that the *fewest empty space first* (FES) policy stochastically minimizes the total-loss process among all nonpreemptive and nonidling policies in the case of single-job processing. However, this is no longer the case in the presence of batch processing. We reexamine a variation of a situation presented in Section 1. Consider the following situation: In a system composed of two parallel queues with buffer sizes $C_1 = 50$ and $C_2 = 100$, respectively, and maximum batch capacity $B = 25$, suppose that at some decision instant the queue lengths are given by $(X_1, X_2) = (1, 25)$. The FES policy would then allocate the server to the first queue which contains a single job, although both buffers are far from overflowing. This decision results in a clear underutilization of the server's processing power. If the second queue was chosen instead, then the server could have formed a full batch of 25 jobs. This example clearly demonstrates the server's dilemma, namely whether to maximize the number of empty spaces in each buffer or maximize its utilization. Moreover, it suggests that a policy attempting to balance these two objectives should perform well.

Therefore, we propose the following *threshold*-type policy. Let T be the threshold level for the number of empty spaces in every queue. Then, if the remaining number of empty spaces in some queue is below T, then the server is allocated to the queue with the fewest empty spaces. If, on the other hand, the number of empty spaces in all queues exceeds the threshold level T, then the server is allocated to the longest queue, in order to maximize its utilization.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

A simulation study was undertaken to compare the performance of the proposed threshold policy against other scheduling policies such as FES, LQ, and nonidling Round Robin (RR), where the latter policy allocates the server to the N queues sequentially and skips the empty queues with zero switching time. The service time distribution for each queue was exponential of rate $\mu = 1$. The arrival process is Poisson with different input intensities ranging from light to medium and then to heavy traffic. Each simulation run had no initialization period, due to the long time horizon used (1 million event epochs). All runs began with a naively given initial state and were terminated when the number of event epochs reached the given prespecified level. Moreover, the systems operating under the various policies were coupled, so that they were given the same arrival processes and the same service time processes.

A related issue was to determine the size of the "optimal" threshold by simulation. We considered systems composed of two, three, and four queues for various arrival rates and various batch and buffer sizes. A fairly large number of threshold levels were examined in each case (e.g., for the third scenario, we used $T = 2,4,6,8$; for the fourth one, $T = 5,10,15,20,25,30,35,40,45$). The results of these simulations are summarized in Table 1 (average over 100 runs, with 1 million event epochs per run). All the results suggest that the best threshold level is given by $T = \min_i\{C_i\} - B$.

The results of the simulation study that compares the various policies for systems composed of two queues are shown in Figure 2 (average over 100 runs). The top left panel in Figure 2 corresponds to a system in which one of the queues has a very small number of buffer places. It can be seen that in the presence of light traffic (low total arrival rate to the system), the FES policy outperformed the remaining scheduling policies in terms of jobs lost due to buffer overflows and, consequently, in terms of jobs departed from the system (due to the coupling of arrival and service time processes employed in the simulation). It is worth noting that no large differences between the "optimal" threshold (TH-1, $T = 1$) and FES policies were observed. On the other hand, the second threshold policy (TH-0, $T = 0$) considered, and, in particular, the RR and the LQ policies, clearly underperformed the remaining policies. However, in the presence of heavy traffic, the two threshold policies performed considerably better in terms of lost jobs (and in terms of departures) than the

Table 1. Determination of Optimal Threshold Level

Buffer Capacity	Batch Sizes	Best Threshold
$C = [3\ 10]$	$B = 2$	$T = 1$
$C = [8\ 10]$	$B = 3$	$T = 5$
$C = [10\ 20]$	$B = 4$	$T = 6$
$C = [50\ 100]$	$B = 25$	$T = 25$
$C = [5\ 10\ 20]$	$B = 3$	$T = 2$
$C = [12\ 18\ 20\ 30]$	$B = 7$	$T = 5$

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

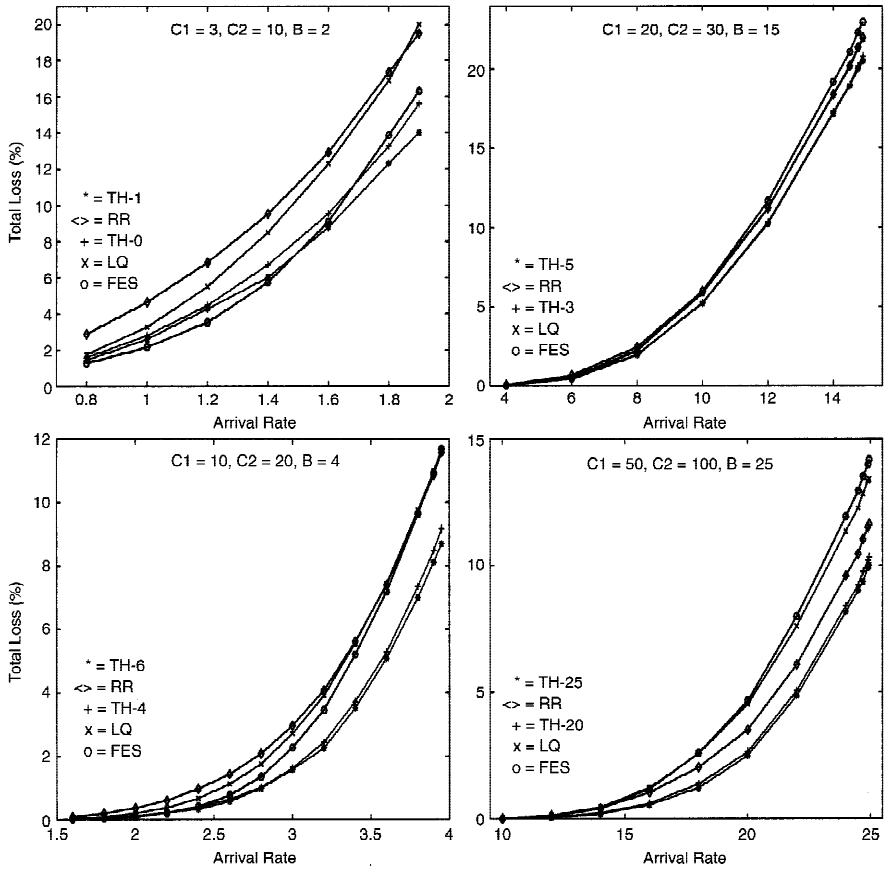


FIGURE 2. Simulation results for two queues.

other three policies. Moreover, the larger the arrival rate, the larger the improvement in their performance.

For systems with larger buffer spaces (remaining three panels in Fig. 2), the threshold policies outperformed the other policies in heavy traffic, although for light traffic, the performance of all the policies under consideration was essentially identical. However, the "optimal" threshold policy (i.e., TH-6, TH-5, and TH-25) outperformed its threshold competitor (i.e., TH-4, TH-3, and TH-20, respectively), although the differences could be characterized as marginal. It is worth noting that the LQ policy exhibited the worst overall performance, whereas the performance of the RR policy improved as the number of buffer spaces increased (compare, in particular, the two bottom panels).

In Figures 3 and 4, comparisons of the various policies for larger systems—involving up to 20 queues—with a much more unbalanced distribution of buffer sizes are presented. The results are fairly similar with those obtained from two-

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

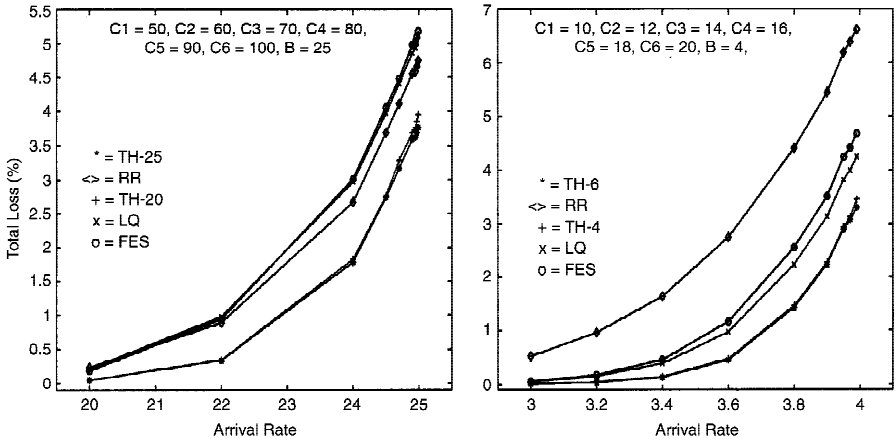


FIGURE 3. Simulation results for six queues.

queue systems. However, it is worth noting the inferior performance of the RR for a larger system with fairly small buffers (see the right-hand panel of Fig. 3); similarly, when the size of the threshold starts deviating markedly from the optimal, the performance of a threshold policy deteriorates significantly, although it still outperforms the nonthreshold-based policies.

5.1. Performance Evaluation of the Threshold Policy

In this subsection, we examine the performance of the proposed threshold policy. In order to keep things simple, we examine the case of two queues, but the methodol-

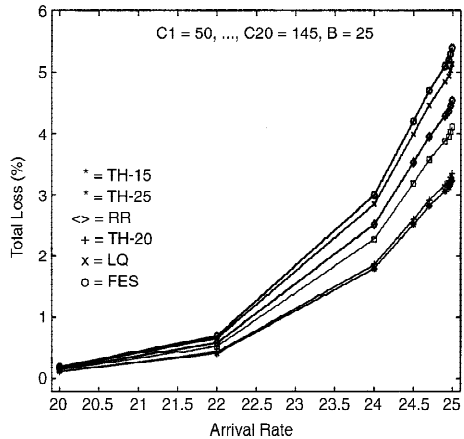


FIGURE 4. Simulation results for 20 queues.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

ogy generalizes to handle an arbitrary number of queues in a straightforward manner. We introduce a *fluid* model, where the queues correspond to reservoirs of size C_i, $i = 1, 2$, containing fluid and the batch capability of the server to a bucket of size B that is used to empty them. Let λ_i denote the *rate* at which liquid flows into reservoir i, t_i the time required to empty reservoir i, c_i the cost per unit volume for fluid lost from reservoir i, and x_i (y_i) the volume of fluid in reservoir i at the present (next) decision epoch.

Here, we focus on stationary policies that each control decision depends only on the *current* state information. The problem can then be formulated using the standard stochastic dynamic control approach (refer to, e.g., [10]).

Let $V(x_1, x_2)$ be the α-discounted cost function under optimal control with respect to initial reservoir levels $\vec{\mathbf{x}} = (x_1, x_2)$. At the initial decision epoch, if we choose to empty the fluid from reservoir 1, the dynamics are

$$y_1 = ([x_1 - B]^+ + \lambda_1 t_1) \wedge C_1, \tag{32}$$

$$y_2 = (x_2 + \lambda_2 t_2) \wedge C_2. \tag{33}$$

The amount of fluid lost from the system is given by

$$([x_1 - B]^+ + \lambda_1 t_1 - C_1)^+ + (x_2 + \lambda_2 t_2 - C_2)^+. \tag{34}$$

Analogous expressions can be written if we decide to drain fluid from reservoir 2 instead. We then have

$$V(\vec{\mathbf{x}}) = \min_{i=1,2,\, j\neq i} \{c_i([x_i - B]^+ + \lambda_i t_i - C_i)^+ + c_j(x_j + \lambda_j t_j - C_j)^+ + \exp(-\alpha t_i) V(\vec{\mathbf{y}}_i(\vec{\mathbf{x}}))\}, \tag{35}$$

where $\vec{\mathbf{y}}_1(\vec{\mathbf{x}}) = (y_1, y_2)$ (see (32)–(33)) and analogously for $\vec{\mathbf{y}}_2(\vec{\mathbf{x}})$. Corresponding to the discrete model we considered in earlier sections, we can assume that $\lambda_i = \lambda$, $c_i = 1$, $i = 1,2$, and $t_1 = t_2 = 1$, because service times are independent of the buffer sizes. Problem (35) can be solved by the standard successive approximation method by setting $V(\vec{0}) = 0$ and the discount factor to a value very close to 1 [e.g., $\exp(-\alpha) = 0.999$]. The numerical results obtained for a number of cases, including those presented in Table 1 involving two queues, are in complete agreement with our simulation results.

Remark 5.1: There are other ways to investigate the performance of the threshold policies for systems with finite buffers of unequal sizes. For example, one can formulate the corresponding problem as a semi-Markov decision problem [8] and then proceed to establish properties of the optimal scheduling policy. However, the fluid formulation presented above captures all of the essential features of the system under consideration, but still remains fairly straightforward to analyze and identify the optimal rule.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

6. CONCLUDING REMARKS

In this article, the problem of dynamic allocation of a server with batch-processing capability to incompatible job classes is studied. The main results derived in this work can be summarized as follows:

- Under symmetric loading and all buffers having infinity capacity, it is shown that the Longest Queue (LQ) first policy maximizes the system's aggregate throughput.
- When all buffers are finite and of *equal* sizes, the LQ policy is equivalent to the Fewest-Empty-Space queue first policy, which is shown to stochastically minimize the total losses due to buffer overflows.
- When all buffers are finite but of *unequal* sizes, there is a nontrivial decision to be made that balances the immediate benefits of fully utilizing the server capacity with the potential danger of traffic loss caused by overflow due to limited buffer space. Through simulation studies, we identify a threshold-type policy that consistently outperforms other scheduling policies that have been proposed in the literature; the optimal threshold level is given by $T = \min_i\{C_i\} - B$. The results are also verified in the heavy-traffic regime by evaluating the performance of a fluid model under the threshold policy using dynamic programming arguments.

There are several interesting issues for further investigations; among them are (i) whether the LQ and/or the FES policies continue to be optimal for the model under investigation for general service time distributions and (ii) if, in addition, *preemptive* service disciplines are allowed.

Acknowledgments

The authors would like to thank an anonymous reviewer for pointing out an important omission in the original proof of Lemma 2.3 and for the helpful comments and suggestions.

The work of G. Michailidis and N. Bambos was supported by the National Science Foundation. This material is based upon work supported by, or in part by, the U.S. Army Research Office under contract/grant number DAAH04-94-G-0214.

References

1. Avramidis, A.N., Healy, K.J., & Uzsoy, R. (1998). Control of a batch processing machine: A computational approach. *International Journal of Production Research* 36: 3167–3181.
2. Deb, R.K. & Serfozo, R.F. (1973). Optimal control of batch service queues. *Advances in Applied Probability* 5: 340–361.
3. Duenyas, I. & Neale, J.J. (1997). Stochastic scheduling of a batch processing machine with incompatible job families. *Annals of Operations Research* 70: 191–220.
4. Koole, G. & Righter, R. (2001). A stochastic batching and scheduling problem. *Probability in the Engineering and Informational Sciences* 15: 465–479.
5. Koole, G., Sparaggis, P.D., & Towsley, D. (1999). Minimizing response times and queue lengths in systems of parallel queues. *Journal of Applied Probability* 36: 1185–1193.
6. Lindvall, T. (1993). *Lectures on the coupling method*. New York: John Wiley & Sons.
7. Nain, P., Tsoucas, P., & Walrand, J. (1989). Interchange arguments in stochastic scheduling. *Journal of Applied Probability* 27: 815–826.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press

8. Putterman, M.L. (1994). *Markov decision processes*. New York: John Wiley & Sons.
9. Righter, R. & Shanthikumar, J.G. (1992). Extremal properties of the FIFO discipline in queueing networks. *Journal of Applied Probability* 29: 967–978.
10. Ross, S. (1983). *Introduction to stochastic dynamic programming*. New York: Academic Press.
11. Shaked, J. & Shanthikumar, J.G. (1994). *Stochastic orders and their applications*. New York: Academic Press.
12. Sparaggis, P.D., Cassandras, C.G., & Towsley, D. (1993). On the duality between routing and scheduling systems with finite buffer space. *IEEE Transactions on Automatic Control* 38(9): 1440–1446.
13. Stoyan, D. (1983). *Comparison methods for queues and other stochastic models.* New York: John Wiley & Sons.
14. Towsley, D. (1995). Application of majorization to control problems in queueing systems. In P. Chretienne et al. *Scheduling Theory and its Applications*. New York: John Wiley & Sons.
15. Uzsoy, R. (1995). Scheduling batch processing machines with incompatible job families. *International Journal of Production Research* 33: 2685–2708.
16. Walrand, J. (1988). *An introduction to queueing networks*. Englewood Cliffs, NJ: Prentice-Hall.
17. Wasserman, K.M. & Bambos, N. (1996). Optimal server allocation to parallel queues with finite-capacity buffers. *Probability in the Engineering and Informational Sciences* 10: 279–283.
18. Winston, W. (1977). Optimality of the shortest line discipline. *Journal of Applied Probability* 14: 181–189.

https://doi.org/10.1017/S0269964802163029 Published online by Cambridge University Press