The Annals of Applied Probability
1998, Vol. 8, No. 3, 822–848

HEAVY TRAFFIC ANALYSIS OF A SYSTEM WITH PARALLEL SERVERS: ASYMPTOTIC OPTIMALITY OF DISCRETE-REVIEW POLICIES

BY J. MICHAEL HARRISON

Stanford University

This paper is concerned with dynamic scheduling in a queueing system that has two independent Poisson input streams, two servers, deterministic service times and linear holding costs. One server can process both classes of incoming jobs, but the other can process only one class, and the service time for the shared job class is different depending on which server is involved. A bound on system performance is developed in terms of a single pooled resource, or super-server, whose capabilities combine those of the original two servers. Thereafter, attention is focused on the heavy traffic regime, where the combined capacity of the two servers is approximately equal to the total input rate. We construct a discrete-review control policy and show that if its parameters are chosen correctly as one approaches the heavy traffic limit, then its cost performance approaches the bound associated with a single pooled resource. Thus the discrete-review policy is proved to be asymptotically optimal in the heavy traffic limit. Although resource pooling in heavy traffic has been observed to occur in other network scheduling problems, there have been very few studies that rigorously proved the pooling phenomenon, or that proved the asymptotic optimality of a specific policy. Our discrete-review policy is obtained by applying a general method, called the BIGSTEP method in an earlier paper, to the parallel-server model.

1. Introduction. Consider the stochastic processing system portrayed schematically in Figure 1. The circles in this figure represent two processing resources, or servers, and the open-ended rectangles represent buffers in which jobs of two different classes reside. Jobs of class k arrive according to a Poisson process at an average rate of λ_k per hour, and the two input streams are assumed to be independent. As indicated in Figure 1, we take the average arrival rates to be $\lambda_1 = 1.3\rho$ and $\lambda_2 = 0.4\rho$, where ρ is a parameter that will be allowed to vary. Each job requires a single service before it departs, and class 1 can be processed by either server 1 or server 2, whereas class 2 can be processed only by server 2. Jobs of each class remain in buffer storage until they are withdrawn for service; it is not necessary to decide which server will handle a class-1 job until the service actually begins. For concreteness, we assume that a service must be completed without interruption once it has begun. Using the terminology proposed in [7], we identify three different *processing*

Received October 1996; revised October 1997.
AMS 1991 *subject classifications*. Primary 60K25, 90B15, 90B22.
Key words and phrases. Queueing theory, heavy traffic, BIGSTEP method, resource pooling.

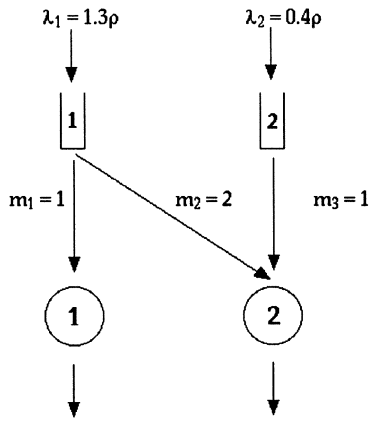

FIG. 1. *Two servers working in parallel.*

activities, as follows:

activity 1 = processing of class-1 jobs by server 1,

activity 2 = processing of class-1 jobs by server 2,

activity 3 = processing of class-2 jobs by server 2.

To make things simple, let us assume that service times for activity j are deterministically of length m_j hours, where $m_1 = m_3 = 1$ and $m_2 = 2$, as shown in Figure 1. That is, it takes exactly one hour for server 1 to process a job of class 1 or for server 2 to process a job of class 2, but it takes two hours for server 2 to process a job of class 1.

If ρ is near 1, the parallel-server system is "in heavy traffic," meaning that its overall input rate and overall processing capacity are approximately equal. To see that this is true, consider the case $\rho = 1$. Devoting all of its time to class 1, server 1 can handle only one of the 1.3 class-1 jobs that arrive per hour on average, and server 2 must spend 60% of its time in processing the class-1 jobs that are left over (0.3 jobs per hour $\times$ 2 hours per job = 0.6). Server 2 has 40% of its time left for class-2 jobs, which is just adequate to handle the arrival rate of $\lambda_2 = 0.4$. In this paper, we focus on system behavior in the heavy traffic regime, and more particularly, on dynamic scheduling in the heavy traffic regime.

To complete the model specification, let us assume that holding costs are continuously incurred at a rate of h_k dollars per hour for each class-k job that remains within the system, with the specific numerical values

$$h_1 = 3 \quad \text{and} \quad h_2 = 1. \tag{1}$$

Each time a service is completed, the system manager must decide which job class the newly freed server will process next, if there is in fact a choice, or the manager can choose to keep the server idle if he or she chooses; a similar

scheduling decision must be made each time a new job arrives to find one or both servers idle. In making these scheduling decisions, the manager strives to minimize holding costs.

Following standard practice in queueing theory, let us denote by $\mu_j = 1/m_j$ the average service rate for activity j, and further define

$$c_1 = h_1 = 3, \qquad c_2 = h_1 = 3 \qquad \text{and} \qquad c_3 = h_2 = 1. \tag{2}$$

Thus c_j is the holding cost rate for the job class that is served in activity j. When the system contains q_1 jobs of class 1 and q_2 jobs of class 2, holding costs are continuously incurred at a total rate of $h_1 q_1 + h_2 q_2$, and activity j decreases this total at an average rate of $c_j \mu_j$ per hour spent in the activity. Guided by the classical $c\mu$ rule of scheduling theory, one might plausibly adopt the following *greedy scheduling rule*. First, server 1 should obviously spend as much time as it can processing class 1, going idle only when buffer 1 is empty. Second, when server 2 completes the processing of a job and finds new jobs waiting in both buffer 1 and buffer 2, it should choose between activities 2 and 3 so that the chosen activity j maximizes $c_j \mu_j$. (If there are waiting jobs in only one buffer, the server will presumably take one of them, and if both buffers are empty it must go idle.) Since $c_2 \mu_2 = 3/2$ and $c_3 \mu_3 = 1$, this means that server 2 will always choose to serve class 1 when confronted with a choice.

System behavior under the greedy scheduling rule has been simulated for the case $\rho = 0.95$, and the results are presented in Figure 2, which shows the simulated queue lengths for class 1 and class 2 (including any jobs that may be in service) as a function of time. One sees from this plot that the greedy rule is disastrously ineffective. In fact, the system fails to achieve a statistical equilibrium: the queue length for class-2 jobs grows in roughly linear fashion as a function of time, and upon reflection, the source of this instability becomes obvious. Server 2, eager to reduce holding costs as quickly as possible, allocates too much time to class-1 jobs when there are class-1 jobs in the system, which leaves it with too little leftover capacity to handle the flow of class-2 jobs, and also leaves server 1 with nothing to do some of the time. In 5000 hours of simulated operation, server 2 spent 71% of its time on class-1 jobs, which did not leave that server with enough residual capacity to handle the demands of class 2, while server 1 experienced 13% idleness. These results show that dynamic scheduling of the parallel-server system is a problem of some subtlety, although it looks simple at first glance.

One of the frustrations of dynamic scheduling theory for stochastic processing networks is that even for the simplest problems, there typically does not exist a policy described by a few parameters which is exactly optimal. Consider our parallel-server problem, for example, with either a long-run average cost criterion or an infinite-horizon discounted cost criterion. Even if one considers the case of exponential service time distributions and interruptible service (yielding a Markov decision process with minimal state space), there is no reason to believe that a simple optimal policy, characterized by just a few critical numbers, exists. However, lowering aspirations in accordance with the gen-

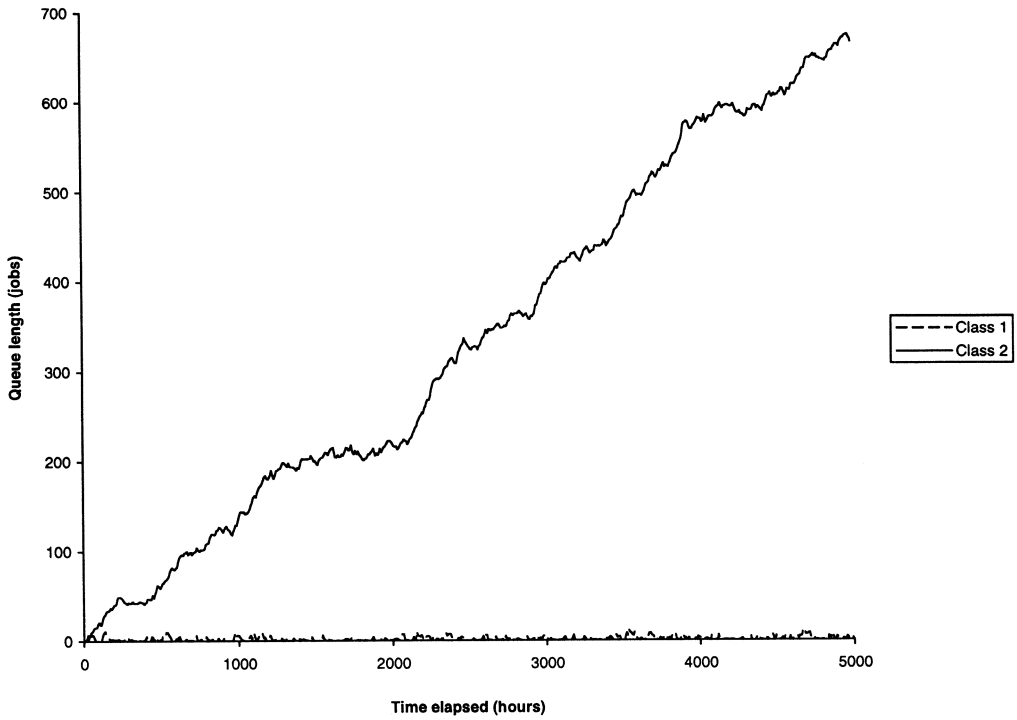

FIG. 2. *System behavior with greedy scheduling* ($\rho = 0.95$).

eral program laid out in [7], one may seek a dynamic scheduling policy that is asymptotically optimal in the heavy traffic limit as $\rho \to 1$. Experience with other relatively simple problems of network control [9, 10, 17, 18] suggests that a policy characterized by a few critical numbers may be asymptotically optimal in heavy traffic, and such results provide both a guide to practical action and qualitative insight into system behavior. Thus we shall undertake a heavy traffic analysis of the parallel-server problem.

To state and explain the significance of the main result proved in this paper, it will be useful to first derive a bound on achievable cost performance in the parallel-server system. Let us denote by $Q_k(t)$ the number of class-k jobs in the system at time t, calling this the class-k queue length process. Also, let $T_j(t)$ denote the total hours devoted to activity j by the associated server during the time interval $[0, t]$. Thus, $I_1(t) = t - T_1(t)$ is the cumulative idleness of server 1 up to time t, and $I_2(t) = t - T_2(t) - T_3(t)$ is the corresponding cumulative idleness process for server 2. Now denoting by $A_k(t)$ the Poisson input process for class k, and assuming hereafter that the system is initially empty, we define "workload processes" W_1 and W_2 for classes 1 and 2, respectively, as follows: $W_1(t) = 2A_1(t) - 2T_1(t) - T_2(t)$ and $W_2(t) = A_2(t) - T_3(t)$. Obviously, $W_2(t)$ represents the total hours of service required from server 2 to complete the processing of all class-2 jobs in the system at time t (recall that only server

2 can process that class), and $W_1(t)$ similarly represents the total hours of service that would be required to complete processing of class-1 jobs in the system at time t *if server* 2 *were required to complete their processing without future help from server* 1. (In making this interpretation, we imagine that if server 1 had already devoted δ hours to the service of a class-1 job, where $0 < \delta < 1$, then server 2 would need $2(1-\delta)$ hours to complete that service.) With a little reflection, the following inequalities should be obvious:

$$Q_1(t) \geq \tfrac{1}{2} W_1(t) \quad \text{and} \quad Q_2(t) \geq W_2(t). \tag{3}$$

Also, defining

$$W(t) = W_1(t) + W_2(t) \tag{4}$$

and

$$X(t) = [2A_1(t) + A_2(t)] - 3t, \tag{5}$$

one can combine the various definitions introduced in this paragraph to conclude that

$$W(t) = X(t) + 2I_1(t) + I_2(t). \tag{6}$$

Now let us define the cost rate experienced at time t,

$$H(t) = h_1 Q_1(t) + h_2 Q_2(t) = 3Q_1(t) + Q_2(t). \tag{7}$$

From (3) and (4), we have that

$$H(t) \geq \tfrac{3}{2} W_1(t) + W_2(t) \geq W(t). \tag{8}$$

Finally, to complete the derivation of our performance bound, we observe that the process $2I_1 + I_2$ on the right-hand side of (6) is nondecreasing and continuous with $I_1(0) = I_2(0) = 0$, while the process W on the left-hand side of (6) is nonnegative. From this, it follows that

$$W(t) \geq Z(t) = X(t) + Y(t), \tag{9}$$

where

$$Y(t) = -\inf_{0 \leq s \leq t} X(s), \tag{10}$$

see [6], Chapter 2. Of course, (8) and (9) together imply that

$$H(t) \geq Z(t) \quad \text{for all } t \geq 0, \tag{11}$$

which is the performance bound referred to earlier. Readers should note that $H(\cdot)$ is the cost rate process associated with an arbitrary scheduling policy, whereas the lower bound $Z(\cdot)$ in (11) is defined directly in terms of model primitives, without reference to any specific policy.

In this paper, we shall consider a sequence of parallel-server systems with $\rho \to 1$, and construct a corresponding sequence of dynamic scheduling policies whose associated cost rate processes $H(\cdot)$ approach the lower bound $Z(\cdot)$ in an appropriate sense. Thus the constructed sequence of policies is said to

be *asymptotically optimal in the heavy traffic limit*. The exact meaning of that phrase will be explained later, but at this juncture two qualitative points deserve emphasis. First, the lower bound $Z(t)$ in (11) holds for every $t \geq 0$ with probability 1, so the sequence of policies constructed here approaches what was called in [7] a *pathwise solution* to the limiting system control problem. That is, the sequence of policies does not just minimize some expected value, but actually minimizes cumulative cost incurred up to any time t, with probability 1, in the heavy traffic limit. In a sense, then, the analysis presented here shows that the parallel-server problem is asymptotically trivial as one approaches the heavy traffic limit, because in general one cannot expect to achieve optimality in such a strong sense.

The second point deserving emphasis is the interpretation of our bounding process Z in terms of a single pooled resource, or super-server. As noted earlier, our definition of the workload process W can be animated as follows: $W(t)$ is the total hours of effort that would be required from server 2, if it were working without assistance, to complete the processing of all jobs in the system at time t, including those partially completed. With this definition of "work," server 2 drains one unit of work from the system per hour that it remains busy, whereas server 1 drains two units of work from the system per hour that it remains busy. The heart of our scheduling problem, of course, lies in the fact that server 1 cannot remain busy unless it has class-1 jobs to work on.

Now suppose that the parallel-server model is modified in the following three ways. First, holding costs are based on the amount of class-1 and class-2 work in the system, rather than on the number of jobs. To be specific, cost is continuously incurred at a rate of \$1.50 per unit of class-1 work in the system, and at a rate of \$1.00 per unit of class-2 work. This means that the holding cost charged for a job being serviced is continuously reduced in proportion to the amount of service it has received, which can only reduce the total cost. Second, the ability of server 1 to complete work at twice the rate of server 2 is extended to class-2 jobs, which means that server 1 can process a class-2 job in half an hour. Finally, servers 1 and 2 are replaced by a single super-server that can drain work from the system at a rate of three units per hour, regardless of whether that work is embodied in jobs of class 1 or class 2. The super-server dominates the combination of server 1 and server 2 in terms of potential cost-performance, because it can focus all service effort on a single job if that is deemed to be desirable, whereas servers 1 and 2 must work on different jobs.

In the modified system described above, the super-server will continue to complete three units of work per hour so long as there are any jobs present. That is, the super-server will fall idle only when its workload backlog reaches zero. It follows that the quantity $Z(t)$ defined by (9) and (10) is the super-server's workload backlog at time t, so the holding cost rate at time t in our modified system must be at least $Z(t)$; that lower bound would be achieved if the super-server could manage to hold all of its backlog in class-2 work, which is not generally possible. Thus $Z(t)$ is a lower bound on the holding cost rate $H(t)$ under any dynamic scheduling policy in our original model.

The result described above might be paraphrased as follows: as one approaches the heavy traffic limit in our parallel-server model, intelligent dynamic scheduling leads to an effective *pooling* of processing resources. This general phenomenon has been noted in other heavy traffic analyses of network scheduling problems that involve alternate routing capabilities, and that literature is admirably summarized in the survey paper by Kelly and Laws [12]. As they explain, studies of resource pooling have been largely heuristic to date. That is, authors use plausible arguments to justify a Brownian approximation to the original control problem under heavy traffic conditions, and then they show how resource pooling occurs in the Brownian approximation. Except for the analysis by Foschini and Salz [3, 4] and Reiman [13] of a very special symmetric problem, the papers surveyed by Kelly and Laws do not prove rigorously that the original and pooled systems are asymptotically equivalent in heavy traffic, and in some cases the authors offer nothing more than tentative guesses as to what sorts of implementable scheduling rules might allow one to approach the heavy traffic performance bounds suggested by the Brownian approximation.

In this paper, we propose a dynamic scheduling policy for the parallel-server problem, one that is characterized by three critical numbers, and prove that if the parameters of that policy are chosen in a certain way, then it achieves the performance bound described earlier in the heavy traffic limit. One contribution of this paper, then, is to provide a complete and rigorous heavy traffic analysis of a dynamic scheduling problem whose solution is not obvious, including a rigorous proof of the resource pooling phenomenon. The policies to be described involve reviewing system status at discrete points in time, and they are obtained by application of a general method, called the BIGSTEP method in an earlier paper [8]. A second contribution of this paper is to provide some evidence in support of the hope expressed in [8] that the BIGSTEP method is generally valid as a mechanical means of deriving scheduling policies which are asymptotically optimal in heavy traffic.

The remainder of the paper is organized as follows. In Section 2, a heavy traffic limit theorem is proved for the super-server workload process Z. In Section 3, we apply the BIGSTEP method to derive a parametric family of discrete-review policies for the parallel-server model. In Sections 4–6, it is shown that parameters of the policy can be chosen so that the associated cost rate process H, properly scaled, converges to the same heavy traffic limit as does the bounding process Z; this establishes the asymptotic optimality of our discrete-review policy. Section 7 presents simulation results for both discrete-review policies and a family of continuous-review policies defined by a single threshold parameter. Some miscellaneous concluding remarks are collected in Section 8, most notably concerning the role of large deviation estimates in our analysis, which are its most novel feature.

Contrary to an intention declared in [8], where heavy traffic behavior of the parallel-server model was described in summary form, this paper does not contain a formal analysis of an "approximating Brownian control problem." That analysis, using the general procedure described in [7] and [11], makes

it plausible that the performance bound discussed above is achievable in the heavy traffic limit, without giving a clear idea as to *how* the bound can be achieved, and it requires a good deal of additional notation that is inessential to our ultimate objectives. The general procedure has now been illustrated in the context of various other network scheduling models [9, 10, 12, 17, 18], so attention will be focused here on construction of an implementable scheduling rule and rigorous proof of its asymptotic optimality. It should also be noted that we use a simplified version of the BIGSTEP method in analyzing the parallel-server model; for reasons explained later, certain aspects of the general method are not needed in this relatively simple setting.

Finally, it must be acknowledged that the analysis undertaken in this paper is quite specific to the structure pictured in Figure 1; it does not suggest how one might manage a more complicated system of parallel servers with overlapping capabilities. For example, if the system had more servers and more input streams, could one still construct a super-server performance bound, what would be the generalized notion of "work" underlying that bound and could the bound be approached in the heavy traffic limit? To achieve complete pooling of all servers in the heavy traffic limit, one obviously needs an adequate degree of overlap in their various capabilities, but the precise mathematical condition is not known at the time of this writing, nor are answers yet available to the other questions raised above. However, by restricting attention to "networks" in which all arrivals require a single service, one eliminates a great deal of the complexity associated with dynamic alternate routing problems, and with that restriction it may indeed be possible to develop a complete and rigorous heavy traffic theory. It is hoped that this paper may serve to stimulate interest in development of such a theory, and to suggest the insights to be gained from it.

2. A heavy traffic performance bound. Following the general procedure laid out in [8], we shall consider a sequence of parallel-server systems indexed by n, each satisfying the assumptions enunciated in Section 1, and such that only ρ changes with n. More specifically, we assume that $\rho(n) \to 1$ fast enough as $n \to \infty$ so that

$$\kappa(n) = n^{1/2}[1 - \rho(n)] \to \kappa \quad \text{(a finite constant)}. \tag{12}$$

The case where ρ approaches 1 from below (so that each system in the sequence is stable, or at least potentially stable) is usually considered most interesting, but strictly speaking, that assumption is not necessary for our purposes. All stochastic processes associated with the parallel-server model depend on n, of course, and that dependence will be indicated by attaching a superscript n to notation established earlier. Thus, for example, $Q_k^n(t)$ is the queue length process for class-k jobs in the nth system, and $H^n(t) = 3Q_1^n(t) + Q_2^n(t)$ is the cost rate process in the nth system.

Recall that three stochastic processes X, Y and Z were defined in terms of our Poisson arrival processes A_1 and A_2 by means of (5), (9) and (10). Viewing all of these processes as random elements of the function space $D[0, \infty)$, one

can express (9)–(10) as

$$Z = \phi(X) \quad \text{and} \quad Y = \psi(X), \tag{13}$$

where (ψ, ϕ) is the *one-sided reflection mapping*, or *one-sided regulator*; see [6], Chapter 2. Returning to our sequence of systems in heavy traffic, let us now define scaled versions of X^n, Y^n and Z^n in the standard way:

$$\begin{aligned} &\tilde{X}^n(t) = n^{-(1/2)} X^n(nt), \qquad \tilde{Y}^n(t) = n^{-(1/2)} Y^n(nt), \\ &\tilde{Z}^n(t) = n^{-(1/2)} Z^n(nt). \end{aligned} \tag{14}$$

The scaling used in (14) is that associated with the central limit theorem (CLT): a scaled process in the nth system (denoted by a tilde) is obtained from a corresponding unscaled process by compressing time by a factor of n and compressing space by a factor of $n^{1/2}$. In similar fashion, the scaled cost rate process in our nth system, using an arbitrary scheduling policy, is

$$\tilde{H}^n(t) = n^{-(1/2)} H^n(nt), \qquad t \geq 0. \tag{15}$$

Applying (13) in the nth system gives $Z^n = \phi(X^n)$ and $Y^n = \psi(X^n)$, and it is well known that the reflection mapping (ψ, ϕ) commutes with linear scaling like that embodied in (14), so from (13) and (14), one has

$$\tilde{Z}^n = \phi(\tilde{X}^n) \quad \text{and} \quad \tilde{Y}^n = \psi(\tilde{X}^n). \tag{16}$$

The program now is to prove a functional central limit theorem for $\tilde{X}^n$, then use (16) and the continuous mapping theorem to obtain a corresponding heavy traffic limit theorem for $\tilde{Z}^n$. For the former, first recall that the mean and variance of a Poisson process are given by $E[A_k(t)] = \text{Var}[A_k(t)] = \lambda_k t$. Thus the translated compound Poisson process X defined by (5) has $E[X(t)] = -3(1-\rho)t$ and $\text{Var}[X(t)] = 4(1.3\rho t) + 0.4\rho t = 5.6\rho t$, and so the scaled version $\tilde{X}^n$ for our nth system has

$$E[\tilde{X}^n(t)] = -3\kappa(n)t \quad \text{and} \quad \text{Var}[\tilde{X}^n(t)] = \sigma^2(n)t, \tag{17}$$

where

$$\sigma^2(n) = 5.6\rho(n). \tag{18}$$

Of course, $\sigma^2(n) \to \sigma^2 = 5.6$ and $\kappa(n) \to \kappa$ as $n \to \infty$ by (12). Thus the following proposition follows directly from the functional central limit theorem for Poisson processes. Here and later, we use the symbol $\Rightarrow$ to denote weak convergence in the function space $D[0, \infty)$ endowed with the standard Skorohod topology, which is equivalent to weak convergence in $D[0, t]$ for each $t > 0$; see [1], Chapter 3.

PROPOSITION 1. *$\tilde{X}^n \Rightarrow X^*$ as $n \to \infty$, where X^* is a Brownian motion with drift parameter -3κ, variance parameter $\sigma^2 = 5.6$ and initial state $X^*(0) = 0$.*

Now the reflection mapping ϕ that carries $\tilde{X}^n$ into $\tilde{Z}^n$ is known to be continuous in the Skorohod topology, so the following is immediate from (16), Proposition 1 and the continuous mapping theorem [1], Section 6.

PROPOSITION 2. $\tilde{Z}^n \Rightarrow Z^* = \phi(X^*)$ *as* $n \to \infty$.

The process Z^* is a reflected or regulated Brownian motion (RBM) with state space $[0, \infty)$, drift parameter -3κ, variance parameter $\sigma^2 = 5.6$ and initial state $Z^*(0) = 0$. Various distributions and expectations associated with Z^* are calculated in [6], especially Chapter 5.

To motivate the development in subsequent sections, let us consider an arbitrary sequence of dynamic scheduling policies for our sequence of systems, and define a corresponding sequence of (unscaled) cumulative cost processes

$$\zeta^n(t) = \int_0^t H^n(s)\, ds, \qquad t \geq 0. \tag{19}$$

As the following argument will show, the appropriate scaled version of (19) is

$$\tilde{\zeta}^n(t) = n^{-(3/2)}\zeta^n(nt) = n^{-(3/2)} \int_0^{nt} H^n(s)\, ds. \tag{20}$$

Making the change of variable $u = s/n$ in (20), and using the definition (15) of the scaled cost rate process $\tilde{H}^n(t)$, it is easy to verify that

$$\tilde{\zeta}^n(t) = \int_0^t \tilde{H}^n(u)\, du. \tag{21}$$

In Section 1, it was shown that $H(\cdot) \geq Z(\cdot)$, and as analogs of (19) and (20), we define

$$\xi^n(t) = \int_0^t Z^n(s)\, ds \tag{22}$$

and

$$\tilde{\xi}^n(t) = n^{-(3/2)}\xi^n(nt) = n^{-(3/2)} \int_0^{nt} Z^n(s)\, ds. \tag{23}$$

Paralleling (21), one has

$$\tilde{\xi}^n(t) = \int_0^t \tilde{Z}^n(u)\, du. \tag{24}$$

The (Riemann) integral mapping that carries $\tilde{Z}^n$ into $\tilde{\xi}^n$ is known to be continuous in the Skorohod topology, so (24) and Proposition 2 together imply that $\tilde{\xi}^n \Rightarrow \xi^*$ as $n \to \infty$, where

$$\xi^*(t) = \int_0^t Z^*(u)\, du. \tag{25}$$

For each fixed t, the random variable $\xi^*(t)$ has a continuous distribution (that is, each $x > 0$ is a continuity point of its distribution function), so one can combine these various observations to obtain the following: for each fixed $t > 0$ and $x > 0$,

$$\limsup_{n\to\infty} P\{\tilde{\zeta}^n(t) \leq x\} \leq P\{\xi^*(t) \leq x\}. \tag{26}$$

In this sense, the scaled cumulative cost process $\tilde{\zeta}^n$ under an arbitrary control policy is *stochastically dominated in the heavy traffic limit* by the process ξ^*.

In the sections that follow, we construct a sequence of discrete-review policies such that the corresponding scaled cost rate processes $\tilde{H}^n$ converge weakly to Z^* as $n \to \infty$, implying that $\tilde{\zeta}^n \Rightarrow \xi^*$ and hence (26) holds with "lim" in place of "lim sup" and "=" in place of "$\leq$." That sequence of policies will be called *asymptotically optimal*, but a stronger term such as "asymptotically dominant" might also be used. It is crucial, of course, that our analysis involves a scaling of time and space under which the bounding processes $\tilde{\xi}^n$ approach a finite but nontrivial limit, for this implies that the *percentage* improvement in cumulative cost achieved by any sequence of policies, relative to our constructed sequence of discrete-review policies, vanishes in the heavy traffic limit. If the scaling were one that gave a trivial limiting performance bound, or an infinite limiting performance bound, no such conclusion could be drawn.

3. Discrete-review policies for the parallel-server model. We return now to the setting of a single model with ρ fixed, so the sequence index n will not appear in this section. Applying the BIGSTEP method [8] to the parallel-server model, we consider discrete-review policies where system status is reviewed at intervals of length l, striving to construct a policy of that type which is nearly optimal in heavy traffic. The only other policy parameters, apart from the period length l, are a pair $\theta = (\theta_1, \theta_2)$ of threshold parameters, or safety-stock values, for the two job classes. Given values of l and θ, the general procedure is as follows. First, system status is reviewed at times $t = 0, l, 2l, \ldots$. At each such time, we observe the current queue length vector $q = Q(t)$, and then solve a deterministic planning problem for the ensuing period. In addition to q, data of the planning problem include the vector θ mentioned above, the vector $\lambda = (\lambda_1, \lambda_2)$ of external arrival rates, the vector $h = (h_1, h_2) = (3, 1)$ of holding cost rates, an input–output matrix R, and a resource consumption matrix A. For our specific problem, R is the 2×3 matrix (one row for each job class and one column for each of the processing activities identified in Section 1)

$$R = \begin{bmatrix} 1 & \frac{1}{2} & \\ & & 1 \end{bmatrix}, \tag{27}$$

while A is the 2×3 matrix (one row for each server and one column for each activity)

$$A = \begin{bmatrix} 1 & & \\ & 1 & 1 \end{bmatrix}. \tag{28}$$

One interprets R_{kj} as the average rate at which activity j decreases the class-k queue length (in jobs per hour) and A_{ij} as the rate at which capacity of server i is consumed by activity j (in hours of server time consumed per hour devoted to the activity.) The deterministic planning problem is a linear program with the following decision variables: x_j is the amount of time that will be devoted to activity j over the coming period by the associated server; u_i is the amount of time that server i will be idle; and z_k is the expected or planned inventory

of class-k jobs at the end of the period. These variables specify a "processing plan" for the coming period. Defining vectors x, u and z in the obvious way, the planning problem can be written as follows: choose x, u and z to

$$\text{minimize} \quad h \cdot z \tag{29}$$

subject to the constraints (here e is a two-vector of ones)

$$z = q + l\lambda - Rx, \qquad u = l\mathrm{e} - Ax, \qquad z \geq \theta, \quad u \geq 0 \quad \text{and} \quad x \geq 0. \tag{30}$$

Of course, the constraint $z \geq \theta$ imposes a lower bound of θ_k on the planned ending inventory z_k, which justifies our previous characterization of θ_k as a "safety-stock" parameter. Our objective (29) is simply to minimize the holding cost rate associated with the planned ending inventory vector z, and it is here that we are using a simplified version of the general BIGSTEP method. In the general method, one must add to the minimand (29) another term of the form $p \cdot u$, where p_i is a penalty rate or "shadow price" associated with idleness of server i. Determination of the coefficients p_1 and p_2 is the most difficult aspect of the BIGSTEP method, but for our parallel-server model it is not necessary to include idleness penalties in the planning problem's objective function, because there is no trade-off between minimizing holding cost and maximizing server utilization. That is, full utilization of both servers is always desirable for minimization of $h \cdot z$, except as idleness may be necessary to meet the threshold requirements on ending inventory ($z \geq \theta$).

The planning problem (29)–(30) involves what might be called a *fluid approximation* to the parallel-server model, meaning that processing activity is treated as deterministic and jobs are treated as a continuous fluid rather than discrete units. Once the meaning of this problem has been grasped, one can solve it by the following stepwise logic. First, because server 1 can process only one job class but server 2 is flexible, we should allocate to server 1 as much class-1 work as possible, subject to the constraints $z_1 \geq \theta_1$ and $u_1 \geq 0$. Recalling that server 1 takes one hour per class-1 job, this means that

$$x_1 = (q_1 + l\lambda_1 - \theta_1)^+ \wedge l. \tag{31}$$

After this allocation is made, there remain $(q_1 + l\lambda_1 - \theta_1 - x_1)^+$ class-1 jobs available for server 2 to process, and server 2 requires two hours to do each of them. Each hour spent on class-1 work reduces holding cost by \$1.50, because $h_1 = 3$, whereas each hour devoted to processing class-2 work reduces holding cost by \$1.00 (the service time is $m_3 = 1$, and $h_2 = 1$). Thus server 2 should allocate as much time as possible to class-1 work, consistent with the constraints $z_1 \geq \theta_1$ and $u_2 \geq 0$, and then allocate as much of its remaining capacity as possible to class 2, subject to $z_2 \geq \theta_2$ and $u_2 \geq 0$. In symbols, this means that

$$x_2 = 2(q_1 + l\lambda_1 - \theta_1 - x_1)^+ \wedge l \tag{32}$$

and

$$x_3 = (q_2 + l\lambda_2 - \theta_2)^+ \wedge (l - x_2). \tag{33}$$

Substituting numerical data for the parallel-server problem in (29) and (30), one can simply verify that (31)–(33) gives one optimal solution. If the initial queue length vector q, together with the vector of expected arrivals $l\lambda$, does not provide enough work to keep both servers busy throughout the period (the threshold requirements on ending inventory are also relevant here), then there may be alternative optima which allow unavoidable idleness to be absorbed by either one server or the other. The specific solution displayed in (31)–(33) is that which maximizes utilization of server 1 when idleness becomes unavoidable.

The following choice of policy parameters will prove to be particularly convenient for theoretical purposes, although other choices might be better in a given practical situation: we shall restrict attention to values of l that are even positive integers, and further take

$$\theta_1 = l\lambda_1 \quad \text{and} \quad \theta_2 = l\lambda_2. \tag{34}$$

That is, the minimum planned ending inventory for each job class is set equal to one period's expected arrivals, which means that the servers must work exclusively on jobs that are already present at the start of the planning period. This choice assures that the linear program (29)–(30) has a feasible solution, and substituting (34) into (31)–(33), our optimal solution reduces to the following:

$$x_1 = q_1 \wedge l, \qquad x_2 = 2(q_1 - x_1) \wedge l, \qquad x_3 = q_2 \wedge (l - x_2). \tag{35}$$

Assuming as always that the system is initially empty, we have $q = 0$ at the initial review point $t = 0$. Then (35) reduces to $x_1 = x_2 = x_3 = 0$, meaning that both servers are obliged to remain idle throughout the first review period. At the end of the first period, whatever queue length vector q may be observed, the time allocations x_1 and x_3 dictated by (35) are integers, and the time allocation x_2 is an even integer (remember that only even integer values of l are to be considered). Because all service times are deterministic, with $m_1 = m_3 = 1$ and $m_2 = 2$, each of our time allocations x_j translates into an integer number of jobs to be processed during the coming period, and all of those jobs are on hand at the start of the period.

Thus there is no uncertainty or ambiguity about how the solution (35) is to be implemented, and there will be no services partially completed when the period ends. The same story is repeated in subsequent periods. Finally, recalling that our idleness variables u_1 and u_2 are defined via $u_1 = l - x_1$ and $u_2 = l - (x_2 + x_3)$, it is easy to verify that the time allocations (35) give

$$u_1 = (q_1 - l)^- \quad \text{and} \quad u_2 = [(2q_1 + q_2) + 2u_1 - 3l]^-. \tag{36}$$

4. Further identities. Throughout this section, we continue to analyze a single parallel-server model with ρ fixed (thus the sequence index n will not appear), operating under the discrete-review policy described in Section 3. The length l of review periods is assumed to be an even integer, which simplifies matters as just explained, and server time allocations x_j for a given period are

defined in terms of l and the initial queue length vector $q = (q_1, q_2)$ by (35). Our initial focus in this section is on discrete-parameter processes associated with the times $t = 0, l, 2l, \ldots$ at which system status is reviewed. As a matter of convention, *period* r ($r = 1, 2, \ldots$) begins at time $(r-1)l$ and ends at time rl. Let $N_k(r)$ be the number of class-k jobs that arrive during period r; these first become eligible for processing at time rl, or equivalently, they first become eligible for processing during period $r+1$. Let $\hat{Q}_k(r)$ be the number of class-k jobs on hand at the beginning of period $r+1$. In terms of earlier notation, $\hat{Q}_k(r) = Q_k(rl)$, and carets will be used similarly in other notation to follow: when both a discrete-parameter and a continuous-parameter version of a process are needed, the discrete-parameter version will be denoted by a caret.

Let $\xi_k(r)$ be the number of class-k jobs that were already on hand at time $(r-1)l$ and remain unprocessed at time rl. (The letter ξ was used in Section 2 with a different meaning, but that earlier meaning will not be needed hereafter.) Thus $\hat{Q}_k(r) = \xi_k(r) + N_k(r)$. Also define: $\hat{B}(r) = 2\xi_1(r) + \xi_2(r)$, the workload *backlog* carried forward from period r to period $r+1$; $\hat{\Delta}(r) = 2N_1(r) + N_2(r)$, the workload increment that arrives during period r and is first eligible for processing during period $r+1$; and $\hat{W}(r) = 2\hat{Q}_1(r) + \hat{Q}_2(r)$, the total workload on hand as period $r+1$ begins. Thus,

$$\hat{W}(r) = \hat{B}(r) + \hat{\Delta}(r) \quad \text{for } r = 1, 2, \ldots . \tag{37}$$

Next, let $U_i(r)$ be the idleness experienced by server i during period r. From the first identity in (36), we have

$$U_1(r+1) = [\hat{Q}_1(r) - l]^- \quad \text{for } r = 1, 2, \ldots . \tag{38}$$

Because $\hat{Q}_1(r) = \xi_1(r) + N_1(r)$, it follows from (38) that

$$U_1(r+1) \le [N_1(r) - l]^- \quad \text{for } r = 1, 2, \ldots . \tag{39}$$

The maximum numbers of class-1 jobs that can be processed by servers 1 and 2 during a period are l and $\frac{1}{2}l$, respectively, and thus the time allocations x_1 and x_2 in (35) imply that

$$\xi_1(r+1) = \left[\hat{Q}_1(r) - \tfrac{3}{2}l\right]^+ \quad \text{for } r = 1, 2, \ldots . \tag{40}$$

Because $\hat{Q}_1(r) = \xi_1(r) + N_1(r)$, we have from (40) that

$$\xi_1(r+1) \le \xi_1(r) + \left[N_1(r) - \tfrac{3}{2}l\right]^+ \quad \text{for } r = 1, 2, \ldots . \tag{41}$$

Next, noting that the total work completed by the two servers during period $r+1$ is $2[l - U_1(r+1)] + [l - U_2(r+1)]$, we have

$$\hat{B}(r+1) = \hat{W}(r) - 3l + 2U_1(r+1) + U_2(r+1) \tag{42}$$

for $r = 1, 2, \ldots$, and combining (42) with the second identity in (36) gives

$$\hat{B}(r+1) = [\hat{W}(r) - 3l + 2U_1(r+1)]^+. \tag{43}$$

Now defining

$$\chi(r+1) = \hat{\Delta}(r) - 3l + 2U_1(r+1) \quad \text{for } r = 1, 2, \ldots, \tag{44}$$

one can use (37) and (44) to rewrite (43) as

$$\hat{B}(r+1) = [\hat{B}(r) + \chi(r+1)]^+ \quad \text{for } r = 1, 2, \ldots . \tag{45}$$

This recursive relationship is of a form familiar in queueing theory; one can use it to express $\hat{B}(\cdot)$ in terms of the partial sums of the random variables $\chi(\cdot)$ as follows. Defining

$$\hat{B}(1) = \chi(1) = 0 \tag{46}$$

as a matter of convention, and letting

$$\hat{S}(r) = \chi(1) + \cdots + \chi(r) \quad \text{for } r = 1, 2, \ldots, \tag{47}$$

it is easy to deduce from (45) that

$$\hat{B}(r) = \hat{S}(r) - \min_{1 \le i \le r} S(i) \quad \text{for } r = 1, 2, \ldots . \tag{48}$$

Let us now define continuous-time versions of the discrete-parameter processes $\hat{\Delta}$, $\hat{B}$ and $\hat{S}$ as follows. First, set $\hat{\Delta}(0) = \hat{B}(0) = \hat{S}(0) = 0$ for completeness, and then, for each $t \ge 0$ satisfying $(r-1)l \le t < rl$ $(r = 1, 2, \ldots)$, let $\Delta(t) = \hat{\Delta}(r-1)$, $B(t) = \hat{B}(r-1)$ and $S(t) = \hat{S}(r-1)$. With these conventions, one has from (48) that

$$B(t) = S(t) - \inf_{0 \le u \le t} S(u) \quad \text{for all } t \ge 0, \tag{49}$$

or equivalently, $B = \phi(S)$ where ϕ is the one-sided reflection mapping introduced in Sections 1 and 2. To bound the difference between B and W, suppose $t \ge 0$ satisfies $(r-1)l \le t < rl$ $(r = 1, 2, \ldots)$ and observe the following: $W(t) \le \hat{W}(r-1) + \hat{\Delta}(r) = \hat{B}(r-1) + \hat{\Delta}(r-1) + \hat{\Delta}(r) = B(t) + \Delta(t) + \Delta(t+l)$; on the other hand, because $3l$ is the maximum amount of work that the two servers can accomplish during a period, $W(t) \ge \hat{W}(r-1) - 3l = B(t) + \Delta(t) - 3l$. Thus one has

$$\sup_{0 \le s \le t} | W(t) - B(t) | \le 3l + 2 \sup_{0 \le s \le t+l} \Delta(s). \tag{50}$$

The last order of business in this section is to establish a relationship between our piecewise-constant process $S(t)$ and the workload netflow process $X(t) = 2A_1(t) + A_2(t) - 3t$ that was introduced in Section 1. If $0 \le t < 2l$, then $S(t) = 0$ by convention, and if $(r-1)l \le t < rl$ for $r = 3, 4, \ldots$, then combining the definition of $\hat{\Delta}(\cdot)$ with (44) and (47) gives

$$S(t) = \sum_{i=1}^{r-2} [\hat{\Delta}(i) - 3l] + 2 \sum_{i=2}^{r-1} U_1(i). \tag{51}$$

Now define a piecewise-constant, deterministic process $\tau(\cdot)$ by setting $\tau(t) = 0$ for $0 \le t < 2l$ and $\tau(t) = (r-2)l$ for $r = 3, 4, \ldots$ and $t \in [(r-1)l, rl)$. The first

term on the right-hand side of (51) is then equal to $X(\tau(t))$, and the second sum is the total idleness experienced by server 1 in periods $2, \ldots, r-1$. Thus we have

$$S(t) = X(\tau(t)) + V(t) \quad \text{for all } t \geq 0, \tag{52}$$

where

$$0 \leq V(t) \leq 2I_1(t). \tag{53}$$

5. Large deviation bounds. As a final preliminary to heavy traffic analysis of our discrete-review control policies, we shall record in this section two probabilistic bounds on the number of class-1 jobs that arrive in a review period, and then explore their implications. Again it will suffice to consider a single system with ρ fixed; the notation established in Section 4 remains in force, with the review period length l viewed as a policy parameter to be determined later. Anticipating the heavy traffic analysis where $\rho \to 1$, and hence $\lambda_1 \to 1.3$, we shall assume hereafter that ρ is close enough to 1 to assure

$$1 < \lambda_1 < \tfrac{3}{2}. \tag{54}$$

Let N_1 denote the number of class-1 jobs arriving during a fixed but arbitrary review period. (This notation is consistent with that used in Section 4.) Thus, N_1 has a Poisson distribution with mean $\lambda_1 l$. The bounds referred to above are the following:

$$P(N_1 \leq l) \leq \exp\{-f(\lambda_1)l\}, \tag{55}$$

where

$$f(x) = x - 1 - \log x \quad \text{for } x \in \left(1, \tfrac{3}{2}\right); \tag{56}$$

and

$$P\left(N_1 \geq \tfrac{3}{2}l\right) \leq \exp\{-g(\lambda_1)l\}, \tag{57}$$

where

$$g(x) = x - \tfrac{3}{2} - \tfrac{3}{2}\log\left(\tfrac{2}{3}x\right) \quad \text{for } x \in \left(1, \tfrac{3}{2}\right). \tag{58}$$

Given (54), it is easy to verify that $f(\lambda_1)$ and $g(\lambda_1)$ are both strictly positive. The large deviation bound (57) is a special use of Markov's inequality: it is obtained by first observing that $P(N_1 \geq \frac{3}{2}l) \leq E[\exp(\theta N_1)]/\exp(\frac{3}{2}\theta l)$ for any $\theta > 0$, and then taking $\theta = -\log(\frac{2}{3}\lambda_1)$ because that choice gives the tightest bound. Similarly, (55) is obtained by first observing that $P(N_1 \leq l) \leq E[\exp(-\theta N_1)]/\exp(-\theta l)$ for any $\theta > 0$, and then taking $\theta = \log \lambda_1$ to get the tightest bound.

Again anticipating the heavy traffic analysis to come, we now fix $T > 0$, let n be a large positive integer, also fixed until further notice, and examine certain aspects of system behavior under our discrete-review policy over the time interval $[0, nT]$. For purposes of this analysis, let us define

$$p = \exp\{-[g(\lambda_1) \wedge f(\lambda_1)]l\}, \tag{59}$$

assuming that l is large enough to ensure $p < 1$. From (55) and (59), it follows that $P(N_1 \le l) \le p$, and this implies

$$E[(N_1 - l)^-] \le lp, \tag{60}$$

because the positive random variable $(N_1 - l)^-$ is bounded by l. Similarly, one has from (57) and (59) that $P(N_1 \ge \frac{3}{2}l) \le p$, and combining this with the strong Markov property of the Poisson process gives

$$E\big[\big(N_1 - \tfrac{3}{2}l\big)^+\big] \le E(N_1)P\big(N_1 \ge \tfrac{3}{2}l\big) \le \lambda_1 lp. \tag{61}$$

The time interval $[0, nT]$ contains all or part of the first K review periods, where K is the smallest integer exceeding nT/l, and (39) gives a bound on the amount of idleness experienced by server 1 during any but the first of those review periods; server 1 is idle for all l hours of the first review period. Thus, combining (39) with (60), and noting that $K \le nT/l + 1$, we have the following:

$$E[I_1(nT)] \le l + (K-1)lp \le l + \frac{nT}{l}lp = l + Tnp. \tag{62}$$

Finally, a bound will be needed for the maximum value achieved by $Q_1(\cdot)$ over the interval $[0, nT]$. To derive this bound, we first observe that, if $(r-1)l \le t < rl$ $(r = 2, 3, \ldots)$, then

$$Q_1(t) \le \hat{Q}_1(r-1) + N_1(r) = \xi_1(r-1) + N_1(r-1) + N_1(r). \tag{63}$$

Using inequality (41) inductively, we have

$$\xi_1(r) \le \sum_{i=1}^{r}\big[N_1(i) - \tfrac{3}{2}l\big]^+ \quad \text{for } r = 1, 2, \ldots, \tag{64}$$

and combining this with (63) gives

$$\sup_{0\le t\le nT} Q_1(t) \le \sum_{i=1}^{K}\big[N_1(i) - \tfrac{3}{2}l\big]^+ + 2\max_{1\le r\le K} N_1(r). \tag{65}$$

Next, observe that

$$\max_{1\le r\le K} N_1(r) \le \tfrac{3}{2}l + \sum_{r=1}^{K}\big[N_1(r) - \tfrac{3}{2}l\big]^+. \tag{66}$$

Because $K \le nT/l + 1$, we have from (61) that

$$E\left\{\sum_{r=1}^{K}\big[N_1(r) - \tfrac{3}{2}l\big]^+\right\} \le K\lambda_1 lp \le nT\lambda_1 p + \lambda_1 lp, \tag{67}$$

and combining this with (65) and (66) yields

$$\begin{aligned} E\Big[\sup_{0\le t\le nT} Q_1(t)\Big] &\le 3(nT\lambda_1 p + \lambda_1 lp) + 2\big(\tfrac{3}{2}l\big) \\ &= 3(nT\lambda_1 p + \lambda_1 lp + l). \end{aligned} \tag{68}$$

6. Heavy traffic performance of discrete-review policies. Let us return now to the sequence of systems described in Section 2, indexed by $n = 1, 2, \ldots$, in which $\rho(n) \to 1$ fast enough to ensure that (12) holds. As in that section, we attach a superscript n to notation previously established in order to indicate a process associated with the nth system, but parameters associated with the nth system are indicated by attaching n as a functional argument. Throughout this section, we consider performance of the nth system operating under a discrete-review policy of the form described in Section 3, so the threshold parameters are $\theta_1(n) = l(n)\lambda_1(n)$ and $\theta_2(n) = l(n)\lambda_2(n)$ as in (34).

To complete the specification of our discrete-review policy for the nth system, we can take $l(n)$ to be any function increasing slowly enough that

$$n^{-(1/2)}l(n) \to 0 \quad \text{as } n \to \infty, \tag{69}$$

but increasing quickly enough that

$$n^{1/2}p(n) \to 0 \quad \text{as } n \to \infty, \tag{70}$$

where $p(n)$ is defined by (59) with $\lambda_1(n)$ in place of λ_1 and $l(n)$ in place of l. Recalling that $\lambda_1(n) \to 1.3$ as $n \to \infty$, one has that $p(n) \sim \exp\{-cl(n)\}$ as $n \to \infty$, where $c = g(1.3) \wedge f(1.3)$. Thus, for example, it would suffice to take

$$l(n) = a \log n \quad \text{where } a > \tfrac{1}{2}c. \tag{71}$$

Obviously, $l(n)$ must grow without bound as $n \to \infty$, but (69) requires the growth to be slow enough that $l(n)$, and hence the threshold parameters $\theta_1(n)$ and $\theta_2(n)$, are small compared with the spatial scale factor $n^{1/2}$ that appears in heavy traffic theorems.

Fixing $T > 0$ hereafter, we shall restrict attention to the time interval $[0, nT]$ in our nth system, which means that all unscaled processes associated with the nth system are viewed as random elements of $D[0, nT]$ and all scaled processes are viewed as random elements of $D[0, T]$. Because T is chosen arbitrarily, weak convergence of the restricted scaled processes in $D[0, T]$ implies weak convergence in $D[0, \infty)$.

Consider first the nondecreasing scaled idleness process $\tilde{I}_1^n(t) = n^{-(1/2)}I_1^n(nt)$, $0 \le t \le T$. From (62), we have

$$E[\tilde{I}_1^n(T)] = n^{-(1/2)}E[I_1^n(nT)] \le n^{-(1/2)}[l(n) + Tnp(n)]. \tag{72}$$

The bound in (72) vanishes as $n \to \infty$ by (69) and (70), implying that

$$\tilde{I}_1^n \Rightarrow 0 \quad \text{as } n \to \infty. \tag{73}$$

Similarly, defining scaled queue length processes $\tilde{Q}_k^n(t) = n^{-(1/2)}Q_k^n(nt)$ in the obvious way, we have from (68) that

$$\begin{aligned} E\Big[\sup_{0 \le t \le T} \tilde{Q}_1^n(t)\Big] &= n^{-(1/2)}E\Big[\sup_{0 \le t \le nT} Q_1^n(t)\Big] \\ &\le n^{-(1/2)}\{3[nT\lambda_1(n)p(n) + \lambda_1(n)l(n)p(n) + l(n)]\}. \end{aligned} \tag{74}$$

The bound in (74) vanishes as $n \to \infty$ by (69) and (70), implying that

$$\tilde{Q}_1^n \Rightarrow 0 \quad \text{as } n \to \infty. \tag{75}$$

For future reference, let us recall (suppressing the sequence index n for the moment) that in Section 4 we defined $\hat{\Delta}_1(r) = 2N_1(r) + N_2(r)$, this representing a workload input increment during the rth review period, and then $\Delta(t)$ was defined as the piecewise-constant extension of $\hat{\Delta}(\cdot)$. The appropriate scaled version of $\Delta(\cdot)$ in our n th system is $\tilde{\Delta}^n(t) = n^{-(1/2)}\Delta^n(nt)$, $0 \le t \le T$, and we shall need the fact that

$$\tilde{\Delta}^n \Rightarrow 0 \quad \text{as } n \to \infty. \tag{76}$$

To prove (76), one first uses the bounds (66) and (67) on the maximum number of class-1 arrivals occurring during any of the first K review periods; those bounds were essential to the proof of (75). To complete the proof of (76), one needs to derive bounds analogous to (66) and (67) concerning the maximum number of class-2 arrivals during any of the first K review periods; because the derivation is virtually identical, details are omitted.

Let us consider now the piecewise-constant extended partial sums process $S(t)$ defined in Section 4. Rewriting the crucial representation (52) in scaled terms for the nth system, one has

$$\tilde{S}^n(t) = \tilde{X}^n(\bar{\tau}^n(t)) + \tilde{V}^n(t), \qquad 0 \le t \le T, \tag{77}$$

where $\tilde{S}^n(t) = n^{-(1/2)}S^n(nt)$ and similarly for $\tilde{X}^n$ and $\tilde{V}^n$, and

$$\bar{\tau}^n(t) = n^{-1}\tau^n(nt), \qquad 0 \le t \le T. \tag{78}$$

(The use of a bar in the notation $\bar{\tau}^n$ emphasizes the presence of scaling associated with functional laws of large numbers, as opposed to processes denoted by tildes, whose scaling is that associated with functional central limit theorems.) It is trivial to show that the deterministic step function $\bar{\tau}^n$ converges to the identity function on $[0, T]$ as $n \to \infty$, and Proposition 1 of Section 2 says that $\tilde{X}^n \Rightarrow X^*$ (a Brownian motion), so one has $\tilde{X}^n \circ \bar{\tau}^n \Rightarrow X^*$ as $n \to \infty$ by the random time change theorem [1], Section 17. Finally, it is immediate from (53) and (73) that $\tilde{V}^n \Rightarrow 0$, so (77) yields

$$\tilde{S}^n \Rightarrow X^* \quad \text{as } n \to \infty. \tag{79}$$

As noted earlier, (49) says that $B^n = \phi(S^n)$. Using the functional central limit theorem (79) and proceeding exactly as in the proof of Proposition 2 (Section 2), one then has

$$\tilde{B}^n = \phi(\tilde{S}^n) \Rightarrow \phi(X^*) = Z^* \quad \text{as } n \to \infty, \tag{80}$$

where $\tilde{B}^n(t) = n^{-(1/2)}B^n(nt)$ as usual. Next, it follows directly from (50), (71) and (76) that $| \tilde{B}^n - \tilde{W}^n | \Rightarrow 0$, and hence (80) implies that

$$\tilde{W}^n \Rightarrow Z^* \quad \text{as } n \to \infty. \tag{81}$$

Because service times are deterministic in our parallel-server model, it follows from the definition of the workload process W (see Section 1) that

$$2[Q_1(t) - 1] + [Q_2(t) - 1] \leq W(t) \leq 2Q_1(t) + Q_2(t). \tag{82}$$

Now (75) says that $\tilde{Q}_1^n \Rightarrow 0$, so (81) and (82) together imply

$$\tilde{Q}_2^n \Rightarrow Z^* \quad \text{as } n \to \infty. \tag{83}$$

Finally, $\tilde{H}^n = 3\tilde{Q}_1^n + \tilde{Q}_2^n$ by definition, so (75) and (83) give the result foreshadowed in Section 2.

PROPOSITION 3. *$\tilde{H}^n \Rightarrow Z^*$ as $n \to \infty$.*

7. A simulation study. Motivated by the theoretical analysis above, a simulation study was undertaken to compare the cost performance of discrete-review policies against the performance bound discussed in Section 1, and also to make comparisons against a natural family of continuous-review policies. The discrete-review policies examined in our simulation study differ somewhat from those discussed in Sections 3–5, which were structured to make the proof of asymptotic optimality as easy as possible. In particular, the discrete-review policies on which we focused in Sections 3–5 force both servers, during each planning period, to work exclusively on jobs present at the start of the period. Such a constraint is obviously unwise in practice, and here we shall make the following pragmatic adjustments. First, both servers remain busy so long as there are jobs available to work on: thus server 1 can be idle only if buffer 1 is empty, and server 2 can be idle only if both buffers are empty. Second, when server 2 completes a service and finds jobs waiting in both buffers, it assigns priority according to the logic described in the next paragraph.

Planning reviews are undertaken at nominal intervals of l hours and, as in Section 3, we require that l be an even integer. The first review point is time zero. Each time a review is undertaken, the next one is scheduled l hours later, but adjustments are made to assure that server 2 is never part way through a service at a review point. That is, if server 2 is busy at the time of a scheduled review (because l is an even integer, this can only happen if server 2 has experienced idleness since the last review), then the next review is delayed until server 2 completes the service in progress. At that point, the queue length vector q is observed (q_1 may include a class-1 job that is currently being processed by server 1) and the planning problem (29)–(30) is then solved to determine a vector x of time allocations for the coming period. In this calculation, the threshold parameter θ_2 is simply set to zero, because there is no motivation to maintain a safety-stock of class-2 jobs in our parallel-server model, and the threshold parameter θ_1 (hereafter called the *target inventory*) is taken to be a positive integer. The only element of x that will actually have operational significance is x_2, for reasons that will become apparent shortly. Nominally, server 2 is directed to serve $\frac{1}{2}x_2$ jobs of class 1 during the coming period, but this need not be an integer in general. Thus we denote by N the

smallest integer greater than or equal to $\frac{1}{2}x_2$, and server 2 gives priority to class 1 during the period that follows until it has served N jobs of class 1; thereafter, it gives priority to class 2.

The traffic intensity parameter was taken to be $\rho = 0.95$ in our simulation study, and an infinite-horizon discounted cost criterion was used. That is, we sought to

$$\text{minimize} \quad ECR = E\left\{\gamma \int_0^\infty e^{-\gamma t} h \cdot Q(t)\, dt\right\}, \tag{84}$$

where γ is the interest rate for discounting. The integral in (84) is the present value of future holding costs, and multiplication by γ converts it to an equivalent uniform cost rate, expressed in dollars per hour. The notation ECR is mnemonic for *expected cost rate*. For the results reported below, the interest rate was fixed at $\gamma = 0.0001$; with time units representing hours, this corresponds to a monthly interest rate (with continuous compounding) of 7.57%. Infinite-horizon discounted costs were estimated using 60,000 hours of simulated operation (preliminary checks showed that costs incurred beyond that point account for less than 1% of the discounted total), and expected values and confidence intervals were based on 10 independent replications of 60,000 hours each.

Figure 3 presents in graphical form the cost performance observed with different combinations of target inventory and review period length. For each choice of period length, the lowest expected cost rate was obtained with a target inventory value between 5 and 10; cost performance degrades quickly below this range, but target inventory values between 10 and 20 are almost as good. For review periods of length $2, 4, \ldots, 16$, Table 1 shows the best target inventory parameter according to our simulation study, as well as our estimate of ECR using that target inventory value, and the upper and lower 95% confidence limits on that estimate of ECR. Our estimates of the minimum achievable expected cost rate increase as the period length increases, but at a rate which is very small and certainly not significant in comparison with the confidence limits displayed in Table 1. Also, Figure 3 shows that our estimates of expected cost rate are nonmonotone as a function of period length for any given target inventory value, but no satisfactory explanation for that nonmonotonicity has been found. The most important conclusions from our simulation study of discrete-review policies are as follows: there is evidence to suggest that shorter review periods are better, in contrast to the emphasis on *long* review periods in our theoretical analysis, but cost performance is insensitive to the period length, and also insensitive to the target inventory value beyond a critical point.

Given the findings above, it is natural to ask what happens as the review period length goes to zero, so our simulation study was extended to consider a one-parameter family of continuous-review policies defined as follows. If server 2, upon completing a service, finds N or more class-1 jobs waiting in buffer 1, then server 2 gives priority to class 1; otherwise, it gives priority to class 2. Beyond that, both servers remain busy as long as there is work for them to do,

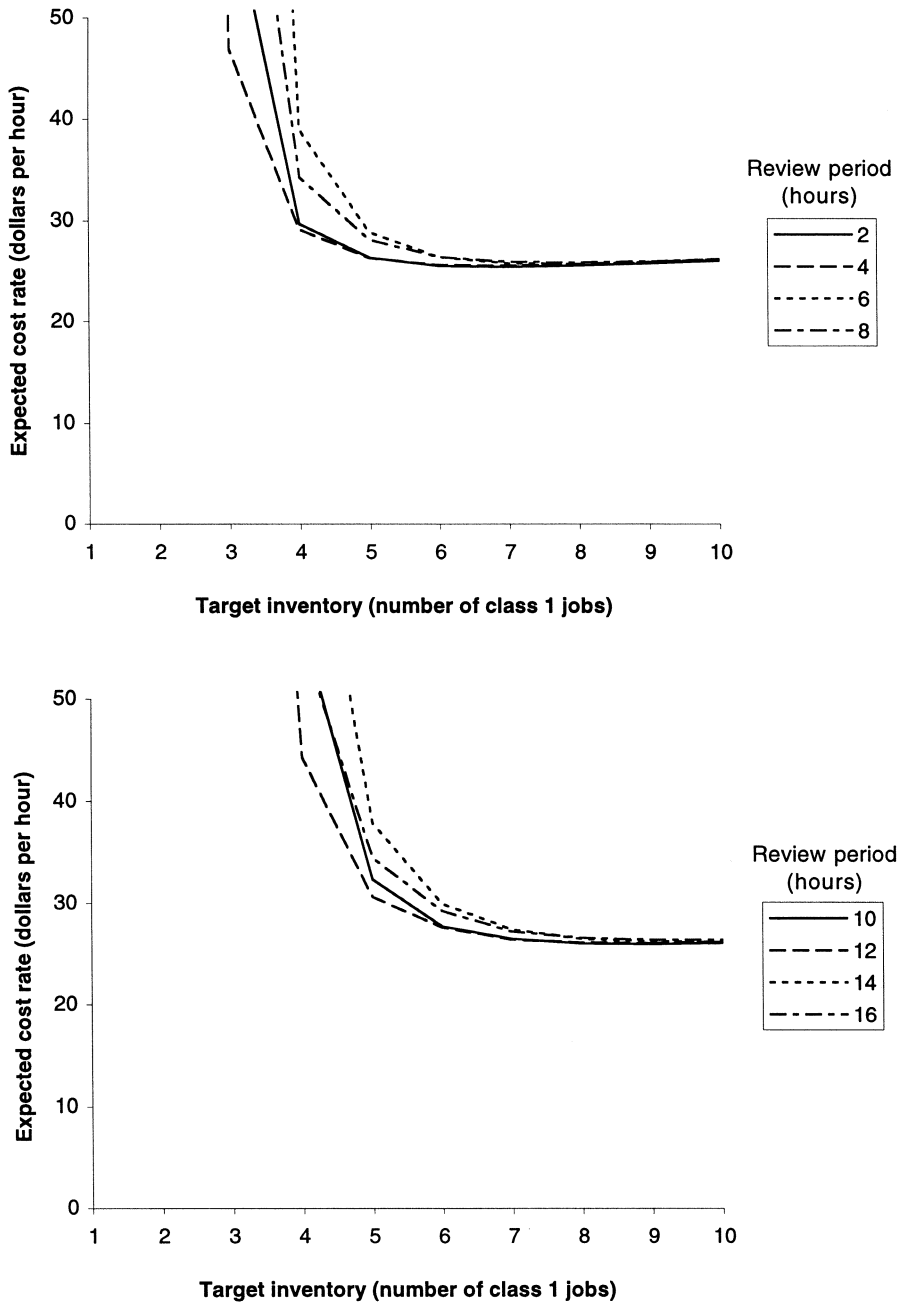

FIG. 3. *Simulation results for discrete-review policies.*

TABLE 1
Simulation results for discrete-review policies

Review period length	2	4	6	8	10	12	14	16
Best target inventory	7	7	8	8	9	9	10	10
Expected cost rate (*ECR*)	25.37	25.53	25.68	25.82	25.96	26.07	26.19	26.37
Lower confidence limit (*LCL*)	24.22	24.38	24.51	24.65	24.80	24.89	25.00	25.13
Upper confidence limit (*UCL*)	26.52	26.68	26.85	26.99	27.12	27.25	27.37	27.61

just as in the discrete-review scenario. We call N the *threshold parameter* of our continuous-review policy. Simulation-based estimates of the expected cost rate (*ECR*) are plotted against N in Figure 4, and the numerical estimates of *ECR* are compiled in Table 2 for $N = 2, \ldots, 9$, along with 95% upper and lower confidence limits on those estimates. (With $N = 1$, our continuous-review policy is equivalent to the greedy scheduling rule described in Section 1.) By a very narrow margin, the lowest estimate of *ECR* is obtained with $N = 6$, but again one sees that cost performance is insensitive to the choice of threshold parameter, at least beyond a critical point.

Table 3 records the best observed cost performance with discrete-review and continuous-review policies, along with two performance benchmarks computed as follows. First, defining the process $Z(t) = X(t) + Y(t)$ exactly as in

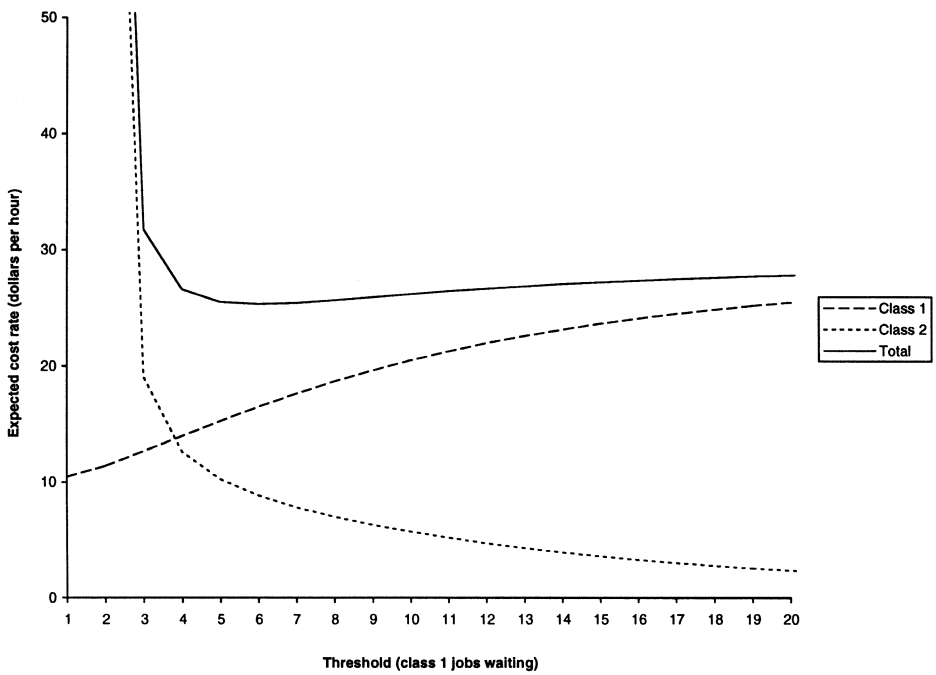

FIG. 4. *Simulation results for continuous-review policies.*

TABLE 2
Simulation results for continuous-review policies

Threshold parameter N	2	3	4	5	6	7	8	9
Expected cost rate (*ECR*)	116.60	31.74	26.60	25.51	25.34	25.44	25.67	25.93
Lower confidence limit (*LCL*)	94.44	29.07	25.12	24.28	24.18	24.31	24.52	24.77
Upper confidence limit (*UCL*)	138.76	34.41	28.08	26.75	26.49	26.57	26.81	27.09

Section 1, let

$$SSPB = E\left\{\gamma \int_0^\infty e^{-\gamma t} Z(t)\, dt\right\}. \tag{85}$$

The notation *SSPB* is mnemonic for *super-server performance bound*: $Z(t)$ is a lower bound on the instantaneous cost rate $H(t) = 3Q_1(t) + Q_2(t)$, as explained in Section 1, implying that (85) provides a lower bound on the expected cost rate (84) under any scheduling policy, and this bound was interpreted earlier in terms of a super-server that combines the capabilities of server 1 and server 2. Now X is a process with stationary, independent increments, or Levy process, and a calculation undertaken in Section 2 can be summarized as follows: $E[X(t)] = -\mu t$ and $\mathrm{Var}[X(t)] = \sigma^2 t$, where

$$\mu = 3(1-\rho) \quad \text{and} \quad \sigma^2 = 5.6\rho. \tag{86}$$

Moreover, any well-behaved Levy process satisfies

$$E[e^{-\eta X(t)}] = e^{\Phi(\eta)t}, \qquad \eta \geq 0, \tag{87}$$

and in our case, the exponent function $\Phi(\cdot)$ is given by

$$\Phi(\eta) = \lambda_1(e^{-2\eta} - 1) + \lambda_2(e^{-\eta} - 1) + 3\eta, \tag{88}$$

where $\lambda_1 = 1.3\rho$ and $\lambda_2 = 0.4\rho$. With $\rho < 1$ (or equivalently, $\mu > 0$), the function $\Phi(\cdot)$ is strictly increasing on $[0, \infty)$ with $\Phi(0) = 0$, and we define $\Psi(\cdot)$ as the inverse of $\Phi(\cdot)$ on $[0, \infty)$. That is, $\Psi(\cdot)$ satisfies

$$\Psi(\Phi(\eta)) = \eta \quad \text{for all } \eta \geq 0. \tag{89}$$

TABLE 3
Cost performance summary

	Value	***LCL***	***UCL***
ECR for best discrete-review policy (period length = 2, target inventory = 7)	25.37	24.22	26.52
ECR for best continuous-review policy (threshold value $N = 6$)	25.34	24.18	26.49
Super-server performance bound (*SSPB*)	17.45	NA	NA
Brownian performance estimate (*BPE*)	17.45	NA	NA

With this notation, one can specialize a general result proved in [5] to obtain the following formula for the super-server performance bound (85):

$$SSPB = \frac{1}{\Psi(\gamma)} - \frac{\mu}{\gamma}. \tag{90}$$

The numerical value of *SSPB* in Table 3 was obtained from (88)–(90) using the data $\rho = 0.95$ and $\gamma = 0.0001$. Finally, the Brownian performance estimate in Table 3 is

$$BPE = E\left\{\gamma \int_0^\infty e^{-\gamma t} Z^*(t)\, dt\right\}, \tag{91}$$

where $Z^* = X^* + Y^*$ is a reflected Brownian motion with drift parameter $-\mu$ and variance parameter σ^2. That is, Z^* is defined in terms of a Brownian motion X^* with parameters $-\mu$ and σ^2, just as Z was defined in terms of X. The analog of (90), derived from the same general result in [5], is

$$BPE = \frac{1}{\Psi^*(\gamma)} - \frac{\mu}{\gamma}, \tag{92}$$

where $\Psi^*(\cdot)$ is the inverse on $[0, \infty)$ of the Brownian exponent function

$$\Phi^*(\eta) = \tfrac{1}{2}\sigma^2\eta^2 + \mu\eta, \qquad \eta \geq 0. \tag{93}$$

The numerical value of *BPE* in Table 3 was obtained from (92) using the data $\rho = 0.95$ and $\gamma = 0.0001$.

The summary figures presented in Table 3 show that the best cost performance achievable with a discrete-review policy is virtually indistinguishable from the best performance achievable with a continuous-review policy, and our super-server performance bound and Brownian performance bound are actually identical to four significant figures, given the parameter values chosen. (The two bounds are not identical in general, however.) Also, the *SSPB* is about 30% smaller than the *ECR* observed with our best scheduling policies. As explained in Section 1, there are three separate factors contributing to the gap between the *SSPB* and actual *ECR* under any given policy. First, the super-server workload process $Z(\cdot)$ is uniformly smaller than the workload process $W(\cdot)$ with two servers working in parallel, regardless of what scheduling policy is used. Second, in computing the *SSPB*, we associate cost with work rather than with jobs, so the cost charged for a job in service is decreased in proportion to the amount of service it has completed. Finally, the *SSPB* corresponds to an ideal scenario where all work is held in the form of class-2 jobs, whereas the average inventory of class-1 jobs is obviously positive under any scheduling rule. A more detailed analysis of our simulation results shows that the last of these factors is by far the most important with the parameter values chosen for the simulation study, accounting for about two-thirds of the 30% gap noted above.

8. Concluding remarks. This paper has analyzed a family of discrete-review scheduling policies that involve safety-stock parameters for each job class. We have considered safety-stock parameters that increase linearly as a function of the review period length l, taking l to be a slowly growing function of the parameter n used to rescale time as a heavy traffic limit is approached. As noted in Section 6, one feasible choice is $l(n) = a \log n$, where a is sufficiently large. Large deviation bounds were then used to prove asymptotic optimality of the proposed scheduling policies. From a technical standpoint, that use of large deviations theory is the most novel feature of the analysis.

The simulation results presented in Section 7 suggest that continuous-review policies of the type described there are at least as good as discrete-review policies, which is hardly surprising. If a sequence of continuous-review policies is to achieve asymptotic optimality, it is undoubtedly essential that the associated threshold parameters grow without bound as heavy traffic is approached, but the threshold parameters must grow more slowly than the spatial scaling factor of $n^{1/2}$ that appears in heavy traffic limit theorems. Kelly and Laws, in Section 2 of their survey paper [12], discuss continuous-review threshold policies for a dynamic routing problem that is roughly similar to ours, and they argue that the threshold parameter must grow at least as fast as $\alpha \log n$ to achieve asymptotic optimality, where α is a certain critical value.

Of course, a proof of asymptotic optimality for continuous-review policies will require more sophisticated methods than those used here. In particular, large deviations theory for *processes* may plausibly provide the key to analysis of continuous-review policies, whereas our analysis of discrete-review policies required only large deviation bounds for Poisson *random variables*.

Acknowledgments. The simulation study described in Section 7 was undertaken by Marcel Lopez, and Jennifer George provided invaluable technical assistance in the preparation of this paper. Thanks are also due to Kingston Duffie for the insights he provided in early conversations about the parallel-server problem, and to Sunil Kumar for correcting a number of errors in a previous draft.

REFERENCES

[1] BILLINGSLEY, P. (1968). *Convergence of Probability Measures*. Wiley, New York.

[2] CHEVALIER, P. B. and WEIN, L. M. (1993). Scheduling networks of queues: heavy traffic analysis of a multistation closed network. *Oper. Res.* **41** 743–758.

[3] FOSCHINI, G. J. (1977). On heavy traffic diffusion analysis and dynamic routing in packet switched networks. In *Computer Performance* (K. M. Chandy and M. Reiser, eds.) 499–514. North-Holland, Amsterdam.

[4] FOSCHINI, G. J. and SALZ, J. (1978). A basic dynamic routing problem and diffusion. *IEEE Trans. Comm.* **COM-26** 320–327.

[5] HARRISON, J. M. (1977). The supremum distribution of a Levy process with no negative jumps. *Adv. in Appl. Probab.* **9** 417–422.

[6] HARRISON, J. M. (1985). *Brownian Motion and Stochastic Flow Systems*. Wiley, New York.

[7] HARRISON, J. M. (1988). Brownian models of queueing networks with heterogeneous customer populations. In *Stochastic Differential Systems, Stochastic Control Theory and Applications* (W. Fleming and P.-L. Lions, eds.) 147–186. Springer, New York.

[8] HARRISON, J. M. (1996). The BIGSTEP approach to flow management in stochastic processing networks. In *Stochastic Networks: Theory and Applications* (F. P. Kelly, S. Zachary and I. Ziedins, eds.) 57–90. Oxford Univ. Press.
[9] HARRISON, J. M. and WEIN, L. M. (1989). Scheduling networks of queues: heavy traffic analysis of a simple open network. *Queueing Systems Theory Appl.* **5** 265–280.
[10] HARRISON, J. M. and WEIN, L. M. (1990). Scheduling networks of queues: heavy traffic analysis of a two-station closed network. *Oper. Res.* **38** 1052–1064.
[11] HARRISON, J. M. and VAN MIEGHEM, J. A. (1997). Dynamic control of Brownian networks: state space collapse and equivalent workload formulations. *Ann. Appl. Probab.* **7** 744–771.
[12] KELLY, F. P. and LAWS, C. N. (1993). Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling. *Queueing Systems Theory Appl.* **13** 47–86.
[13] REIMAN, M. I. (1983). Some diffusion approximations with state space collapse. *Proceedings of the International Seminar on Modelling and Performance Evaluation Methodology, Lecture Notes in Control and Inform. Sci.* **60** 209–240. Springer, Berlin.
[14] REIMAN, M. I. (1984). Open queueing networks in heavy traffic. *Math. Oper. Res.* **9** 441–458.
[15] REIMAN, M. I. and SIMON, B. (1990). A network of priority queues in heavy traffic: one bottleneck station. *Queueing Systems Theory Appl.* **6** 33–58.
[16] VAN MIEGHEM, J. (1995). Dynamic scheduling with convex delay costs: the generalized $c\mu$ rule. *Ann. Appl. Probab.* **5** 809–833.
[17] WEIN, L. M. (1990). Brownian networks with discretionary routing. *Oper. Res.* **38** 1065–1078.
[18] WEIN, L. M. (1991). Scheduling networks of queues: heavy traffic analysis of a two-station network with controllable inputs. *Oper. Res.* **39** 322–340.

GRADUATE SCHOOL OF BUSINESS
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-5015
E-MAIL: harrison_michael@gsb.stanford.edu